Alventis™

Powerful information management
for non-programmers

User's Guide

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Alventis User's Guide

This software is based, in part, on the work of Anders Melander, Gustavo Huffenbacher Daud, and Jordan Russell.

Space images used in Alventis have been generated, authored and/or prepared by NASA and/or STScI using the Hubble Space Telescope. Other parties' images have been created by their respective authors.

With the exception of a few celebrities, the example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person, or event is intended or should be inferred.

Alventis, Alventis Designer, the stylized "AV" Alventis logo, UniGrid, InstaSearch, InfoView, InfoSet, InstaButton, FieldView, MatchBar, QuickStyle, RotoSplitter, UniToggle, PanelZoom, among others, are trademarks or registered trademarks of Alventis Corporation in Canada, the United States, and other countries. Other parties' trademarks or service marks are the property of their respective owners.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Table of Contents

Foreword 0

Part I Introduction 2

1 3 Minute Expert Introduction 2
2 Brief Features Overview 2
3 User's Guide Organization 4
4 Overall Alventis Concept 4

Part II Alventis 9

1 UniGrids 9
2 InfoViews 16
3 Alventis and Designer Interface Overview 19
4 Using Grids 22
5 Grid Summaries 28
6 Memos 30
Memo Introduction 30
Things Unfamiliar 30
Font 33
QuickStyle 34
Paragraph Format 36
Color Picker 39
Bullets and Numbering 41
Symbol Picker 47
Tab Stops 49
Memo Tables 50
Pictures 59
Hyperlinks 62
MemoSearch 66
Find and Replace 72
Style Picker 74
Spellcheck and Thesaurus 75
Drag-and-Drop 79
Other Operations 80
7 Record Styles 81
8 Hidden/Sticky Records 84
9 Style Explorer 85
10 Synopsis Workbench 87
11 Database Explorer 89
12 Security Overview 94
13 Security and Administration 97
14 Server Administration 108
15 Authentication Dialog 113

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

16 Queries List ... 114
17 Query Form ... 116
18 Reports ... 118
19 Search Syntax ... 131
20 Alventis Application Settings ... 133
21 New/Open Record Wizards ... 134
22 Import/Export ... 134
InfoSet Import/Export ... 134
Data Import/Export ... 140
Memo Import/Export ... 149

Part III Designer 152

1 Alventis Designer Overview ... 152
2 FieldViews ... 154
3 InfoView Manipulation Overview ... 160
4 InfoView Editing ... 161
5 InfoView Item Selection and Editing ... 163
6 InfoView Grids ... 170
7 InfoView Items Overview ... 172
8 Layout ... 175
9 Item Formatting ... 186
10 Navigation and Tab Order ... 194
11 Calculated Fields ... 197
12 Validation ... 198
13 Relational Databases Design ... 199
14 New Object Wizards ... 203

Part IV Additional Information 205

1 Server Installation and Setup ... 205
2 Database Principles Overview ... 208
3 Customizing Toolbars and Menus ... 214
4 International Issues ... 221
5 Data Table Verification, Repair, and Backup ... 223
6 System Capacities ... 227

Part V DBISAM SQL Reference 230

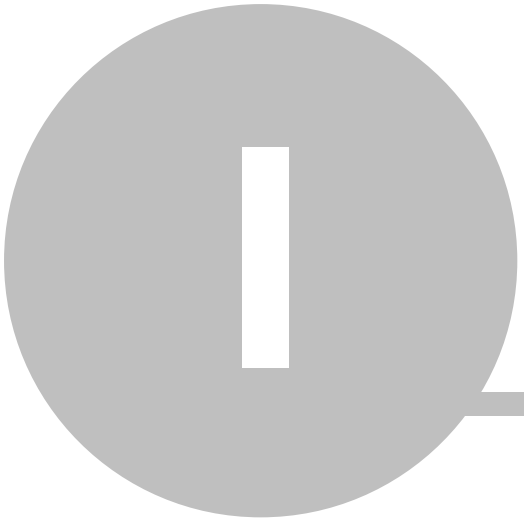
1 Overview ... 230
2 Naming Conventions ... 231
3 Unsupported SQL ... 239
4 Operators ... 240
5 String Functions ... 248

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

6 Numeric Functions ... 254
7 Boolean Functions ... 259
8 Aggregate Functions ... 261
9 AutoInc Functions ... 265
10 Text Search Functions ... 265
11 Data Conversion Functions ... 267
12 Constant Functions ... 270
13 SELECT Statement ... 271
14 INSERT Statement ... 282
15 UPDATE Statement ... 283
16 DELETE Statement ... 286
17 CREATE TABLE Statement ... 289
18 CREATE INDEX Statement ... 294
19 ALTER TABLE Statement ... 295
20 EMPTY TABLE Statement ... 297
21 OPTIMIZE TABLE Statement ... 297
22 EXPORT TABLE Statement ... 298
23 IMPORT TABLE Statement ... 299
24 REPAIR TABLE Statement ... 300
25 UPGRADE TABLE Statement ... 301
26 DROP TABLE Statement ... 301
27 RENAME TABLE Statement ... 302
28 DROP INDEX Statement ... 302
29 START TRANSACTION Statement ... 302
30 COMMIT Statement ... 303
31 ROLLBACK Statement ... 303

Index 304

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Introduction

1 Introduction

1.1 3 Minute Expert Introduction

So, you're an expert in Windows applications and databases and never read manuals? Fine. Here's a one-page overview of only the elements that are unusual or not obvious.

Alventis has UniGrids. Each lists tables from multiple Databases at the top. Perform a search, get some search results as-you-type. Open each result in a standalone InfoView form.

Available data sources are managed in the Database Explorer form.

Each Database's Queries are accessible via the Queries button. Reports are based on these Queries.

Designer is used to create from scratch tables and InfoViews. It has a FieldView form per Database. This lists tables, InfoViews, and fields. 6 fields are standard/System, the rest is up to you. Drag or double-click the fields and drag the tables onto the InfoViews to implement field-based items. You may need to set Rights and Security for newly-created objects if default rights are insufficient or non-existent. Relational Databases are supported.

InstaButtons are unique in that you can have as many of the same kind as you like, and they behave as if they were completely independent. You could have 3 Text Color and 5 Font InstaButtons. Clicking one applies its attributes to the selection. Shift-clicking it typically assigns the selection's attributes to the InstaButton. Drop-down the menu-like dialog to edit the relevant attributes. Color InstaButtons can only be edited but not Shift-assigned.

Splitters that have a little button in their right/bottom end are RotoSplitters. Click the button or Shift-double-click the RotoSplitter to rotate it 90º.

Invoke PanelZoom by double-clicking a Panel or calling the command on the toolbars/menus. The Panel will get zoomed (i.e., maximized) within the form. Note that not all Panels support this feature, but many do.

Memos support a unique MemoSearch feature. Enter the text into one of the Match bar combo boxes, set the marker color next to it, click the marker color button or hit Enter in the combo box: all matches in the Memo are highlighted and shown in the MatchBar next to the scrollbar. Click on it to go to the corresponding place in the Memo.

So, what does your stopwatch say? Don't tell us you weren't timing your reading!

1.2 Brief Features Overview

This chapter should give a quick overview of what Alventis is capable of. Some features pertain to Alventis Designer or Server and are therefore only available in the Professional or Enterprise editions. We'll indicate this as we go. To keep it short, we are omitting lengthy explanations, so this introduction may require some background in Windows applications and databases.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Alventis supports the following elements typical of most database applications:
- Tables, including relationally-linked ones
- InfoView forms based on Tables (Data-entry forms to old-timers)
- A broad array of field types, including Pictures and rich-text Memos
- Queries using a subset of SQL-92
- Fully customizable Reports based on Queries
- Local or Client-Server modes of operation

The central feature of Alventis is Search:
- Searches are performed across all selected Tables regardless of their structure
- Searches rely on full-text indexing to almost instantaneously find all records that meet the search criteria
- Usually, all textual fields are searched, Memos included (the exceptions are fields specifically excluded from searches by the respective Table's designer)
- Searches can use complex expressions and Boolean logic
- Search results from all Tables are presented in a single cumulative list
- You can open records directly from this Search Results list by simply double-clicking them

Alventis Professional adds a "structural editor" called Alventis Designer which extends the above features by providing the following capabilities:
- Create Tables from scratch with unlimited number of fields, and modify existing Tables
- Create InfoView forms from scratch and modify existing ones
- Implement sophisticated InfoView layouts
- All of the above is performed with absolutely no coding or scripting: just drag-and-drop and a few mouse clicks

Alventis Enterprise adds the following features to Alventis:
- Database Server itself, which is a compact standalone application
- Administration of the Servers
- Control over multi-user security, rights, and access privileges

It may not be immediately apparent from all the features listed above, but Alventis is also quite unique in its ability to concurrently and seamlessly integrate multiple sources of information. This means the following:
- Unlimited number of Tables per Database
- Unlimited number of Databases per Server
- Unlimited number of Servers

All of the above is transparent to the data operations, such as searching, viewing, editing, and reporting. In particular, a single list of Search Results may include records from all of the above sources, perhaps even different Servers.
Note that "Server" in the above discussion means either a real Remote Server (a separate application running on some computer) or the local "pseudo-server". This is explained in more detail at the beginning of the Database Explorer chapter.

We will now discuss the available Alventis editions and what each of them offers you.

- Alventis Standard. This includes just the Alventis application itself. You can work with all the existing Tables and InfoViews (together referred to as InfoSets) viewing and editing their data. Aside from these "database user" activities, you can create Queries and base Reports on them.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Default Queries are provided for users who don't know SQL. Designing a Report is about as easy as creating a document, while still offering full flexibility: sorting, multi-level grouping, etc.

- Alventis Professional is essentially Alventis Standard with the addition of Alventis Designer, which is a standalone application. With both tools at your disposal, you can redesign or create from scratch absolutely everything Alventis has to offer, including Tables and InfoView forms. This amounts to a complete development package with results that can rival those of "heavyweight" DBMS platforms.

- Alventis Enterprise is comprised of the Alventis application with Administration capabilities described earlier, the Database Server application, and Alventis Designer. An enterprise-level multi-user installation of Alventis could therefore consist of a single copy of Alventis Enterprise for the Database developer and/or administrator plus the required number of Alventis Standard or Professional copies, one per database user.

1.3 User's Guide Organization

This User's Guide is available to you in several forms, all of which can be downloaded from Alventis web site. No matter which form you choose, they all have much in common:
- They all present the same exact subject matter in the same order
- They are all divided in topics or chapters, each of which attempts to present all relevant information and put it into perspective

On the whole, the Guide is organized more like a book than application Help. In particular, you will not find any single-sentence Help screens whose sole purpose is to answer the question "what is this button?". Answers to such questions, no matter how precise, always tend to result in more questions, such as "so what?". Which is why we chose to write lengthier chapters that try to explain important concepts and ideas first, and only then deal with specific interface elements related to the subject at hand.

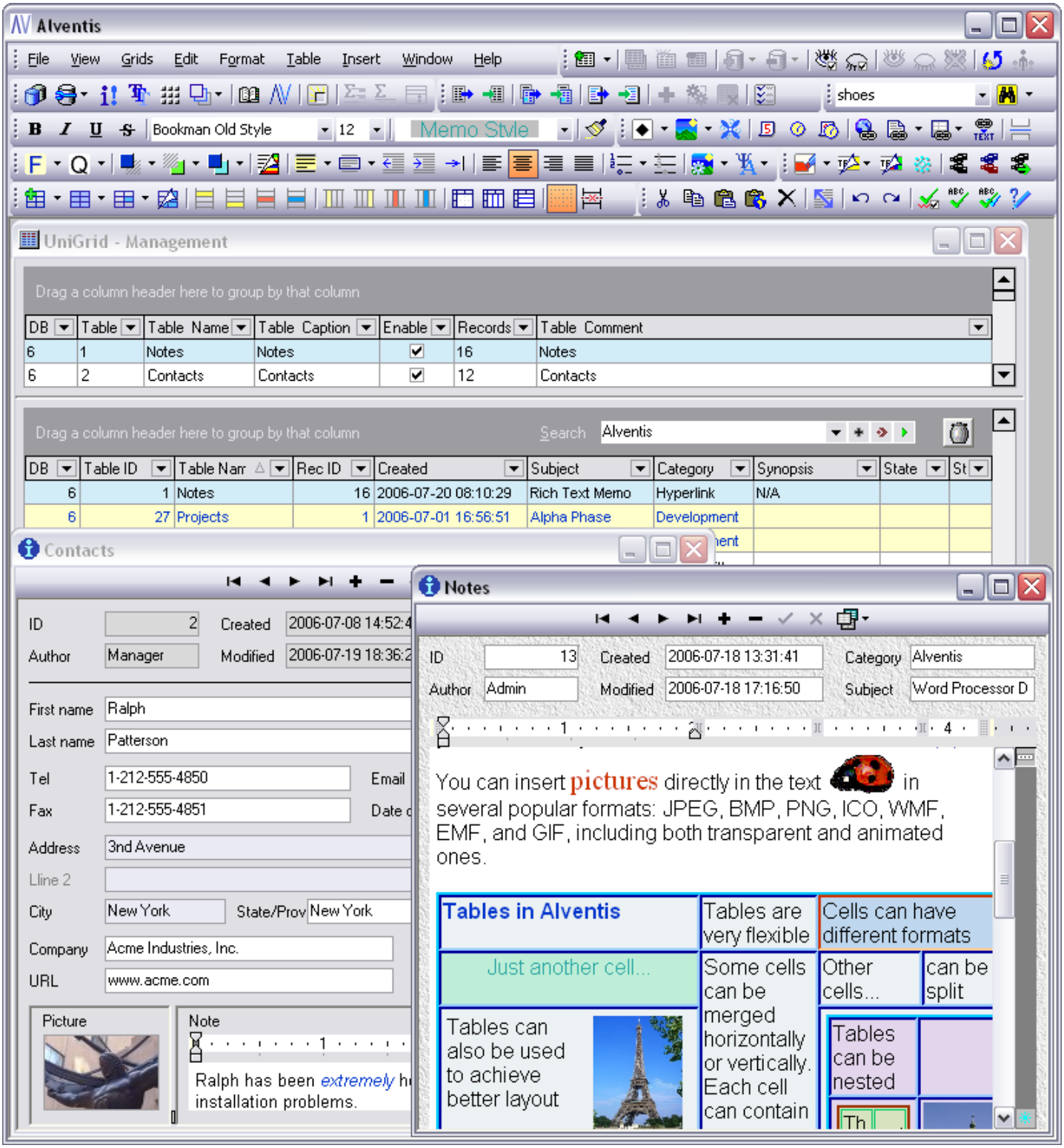
You may want to read this Guide sequentially, in the order in which the topics are listed. Users who are less familiar with basic Principles of Databases may want to read that chapter earlier rather than later. The Table of Contents lists all topics and some sections, so you should be able to find your way.

This is likely the most boring topic of them all, so let's move on to more exciting stuff, shall we?

1.4 Overall Alventis Concept

In this chapter, we will give you a "bird's eye" overview of Alventis.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Alventis
File View Grids Edit Format Table Insert Window Help
shoes
Bookman Old Style
12
Memo Style
UniGrid - Management
Drag a column header here to group by that column
DB Table Table Name Table Caption Enable Records Table Comment
6 1 Notes Notes 16 Notes
6 2 Contacts Contacts 12 Contacts
Drag a column header here to group by that column
Search
Alventis
DB Table ID Table Nam Rec ID Created Subject Category Synopsis State St
6 1 Notes 16 2006-07-20 08:10:29 Rich Text Memo Hyperlink N/A
6 27 Projects 1 2006-07-01 16:56:51 Alpha Phase Development
Contacts
ID 2
Created 2006-07-08 14:52:4
Author Manager
Modified 2006-07-19 18:36:2
First name Ralph
Last name Patterson
Tel 1-212-555-4850
Email
Fax 1-212-555-4851
Address 3nd Avenue
City New York
State/Prov New York
Company Acme Industries, Inc.
URL www.acme.com
Picture
Note
Ralph has been extremely h
installation problems.
Notes
ID 13
Created 2006-07-18 13:31:41
Category Alventis
Author Admin
Modified 2006-07-18 17:16:50
Subject Word Processor D
You can insert pictures directly in the text in several popular formats: JPEG, BMP, PNG, ICO, WMF, EMF, and GIF, including both transparent and animated ones.
Tables in Alventis
Tables are very flexible
Cells can have different formats
Just another cell...
Some cells can be merged horizontally or vertically. Each cell can contain
Other cells...
can be split
Tables can also be used to achieve better layout
Tables can be nested

Alventis is an information management application. It has been designed to:

- Store information in a variety of formats

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

- Enable you to easily locate stored information
- Retrieve, view, and edit said information

When generalized to this extent, Alventis appears - at least at first sight - to be no different from many other Personal Information Managers (PIMs) or Database Management Systems (DBMSs). There are quite a few important differences though.
- Alventis can seamlessly support an infinite number of information formats (i.e., data table structures)
- Alventis can automatically and transparently locate information across all available formats
- Alventis presents the results of a search in a single cumulative list

In other words, Alventis is a database application that can simultaneously support a multitude of data formats, while at the same time giving you as unified a view as possible of all your multi-faceted information.

This is all very nice and general, but a little abstract and theoretical, so let's examine what Alventis can do using an example. While at it, we will try to gradually introduce some useful terms we will be using in the rest of the text.

Let's say we want to keep track of things. "Things" come in a variety of "shapes and sizes" though. We may, for instance, want to put together a list of our friends, their phone numbers, addresses, birthdays, and so on. We may want to make a list of our favorite recipes, what part of the world they are from, what ingredients go into them, and how long their well-blended mix should stay in the microwave. Finally, we would likely want to be able to just jot things down – things that may not fall into any particular category, and may not have any discernable format. Just notes.

Since all of these kinds of information are sufficiently different from one another, it makes sense to keep them separate. We will therefore put each kind in its own data table. Let's call these tables Contacts, Recipes, and Notes. Each of these tables will have some pieces of information in common. It may make sense, for example, to record the date when we recorded a particular Note or Recipe. Or the date we last modified it. Because of this, all tables in Alventis have several standard "System" fields, e.g., Creation Date, Last Modification Date, Subject, Category. Beyond these items, things start to diverge though. Our Contacts table probably needs such fields as: First Name, Last Name, Birthday, Telephone, E-mail, and so on. The Recipes table is likely to record things like: Ingredients, Preparation Procedure, Part of the World. Finally, the Notes table may only need a single item: the Note itself. Such diversity is likely to be the rule for just about any type of information (and table) you can think of. Which is why Alventis can handle tables with any number of fields of almost any kind you may wish for – in addition to the above-mentioned system fields. It is this approach that makes it possible to treat all types of data mostly the same way, while at the same time fully preserving their true "identity", format, etc. Basically: a Contact entry is a record, and a Note is a record, so many operations can regard them as "similar enough". Search, for instance.

Search in Alventis works across all available data formats. Imagine you have a nice database of hundreds of Contacts, dozens of Recipes, lots of Notes, and thousands of records in a bunch of other formats/tables. Imagine now that you are looking for your friend Richard's phone number. Okay, so it's not very likely that you jotted it down somewhere in the middle of a Royale with Cheese Recipe. However, it may be lurking in either the Contacts or the Notes table. Or perhaps some other unforeseen place. This is not a problem with Alventis. Select the tables you want to search in, type your search – and you get a list of all records that contain the name Richard anywhere in them. Doesn't matter which table or which field of that table: Alventis will almost instantaneously locate the requested information and present to you a list of all relevant records. This list gives you a unified view of the search results, even though some records may be coming from the Contacts table, and others may belong to Notes. You can examine each found record in detail by opening it.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Each table in Alventis will usually have at least one form associated with it. We shall refer to such forms as InfoViews. Each InfoView is tailored to display a record from a particular table, so each can have its own presentation format and layout. The Contacts table will likely have an InfoView with quite a few Text Boxes. The Notes table on the other hand may get by with just one large Memo box. Whatever makes sense. Generally speaking, an InfoView is the window through which you view the table it corresponds to. It is here that you can view and navigate records, edit existing, and create new ones. For those of you who have some database or data-processing background, InfoViews are pretty much what some applications refer to as "data-entry forms" or similar.

You can do quite a few other things with records in Alventis. You can perform queries on data tables using the SQL language. You can create Reports and save them or print them (if you still insist on working with paper). You can calculate certain useful things for records presented in grids, such as totals and averages. But at the base of it all lies a deceptively simple idea.
Work with any number of different tables at the same time. Perform searches on them as if they were one. Scroll through search results as if they were coming from the same table. View and edit individual records of all tables in their respective InfoViews.

And this would be the end of our quick presentation – if it weren't for...

Alventis Designer. Its concept is so simple, we'll just say it. It gives you the ability to:
- Create new tables of any kind from scratch
- Modify existing tables by adding, deleting, or changing their fields
- Create or modify InfoView forms

Alventis does come with a number of tables and InfoViews that may be quite useful "out of the box". However, there is nothing special or "built-in" about these items. They have simply been created using Designer – the same exact way you could do it yourself. We just wanted to show you what Alventis can do while giving you something useful at the same time.

The bottom line is: whatever Tables, InfoViews, Queries, and Reports you see in Alventis, you can create all of this stuff yourself – either from scratch or by modifying the existing ones to your liking. All this is accomplished with absolutely no programming or scripting.

In order to give you a general idea of what Alventis is all about, we have omitted many important details. You will find them in the remainder of this manual.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Alventis

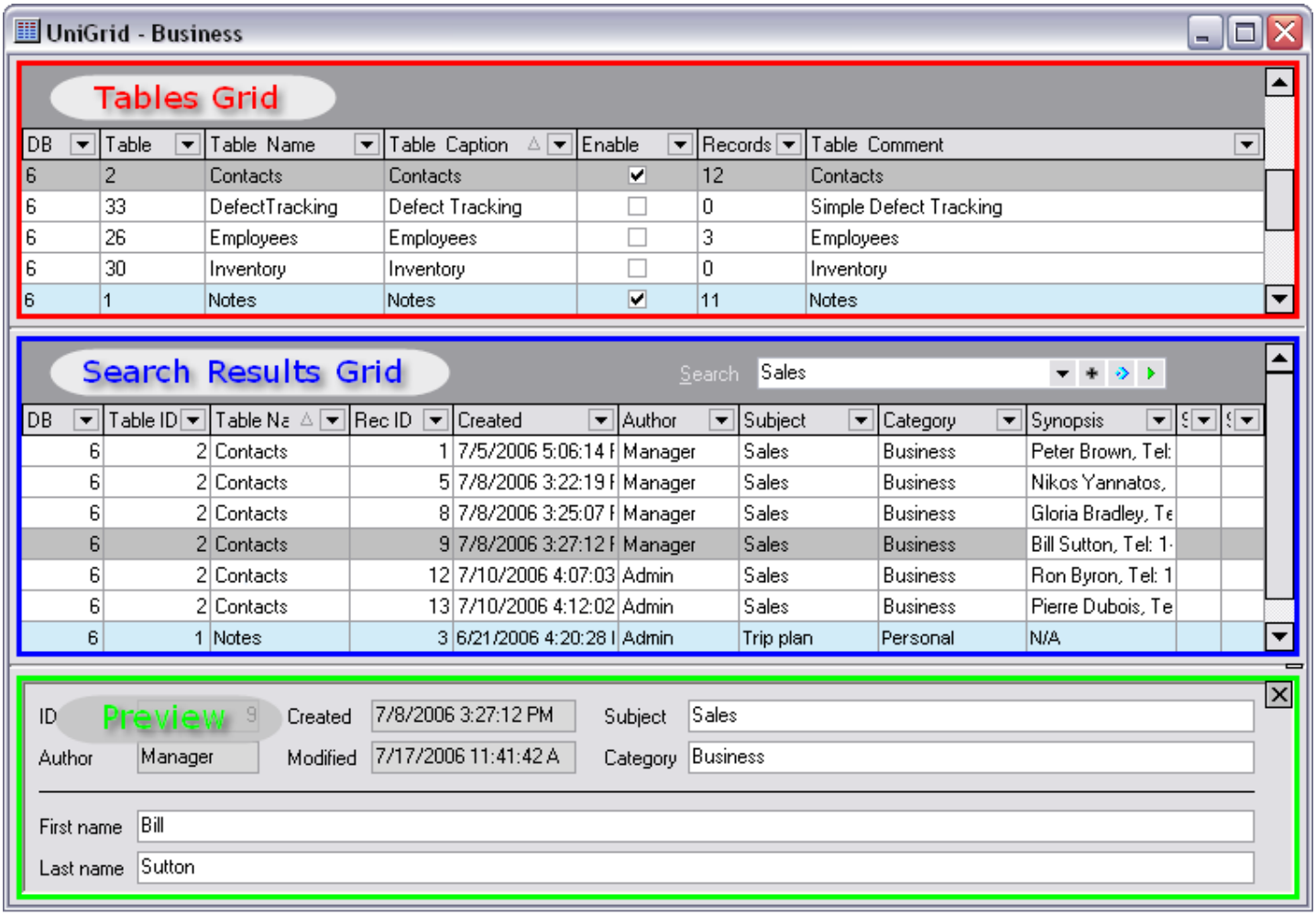
2 Alventis

2.1 UniGrids

Finding and accessing information is perhaps the most important and frequently used operation you are likely to perform in Alventis, and you do so in a form called UniGrid.

A **UniGrid** has 3 panes:

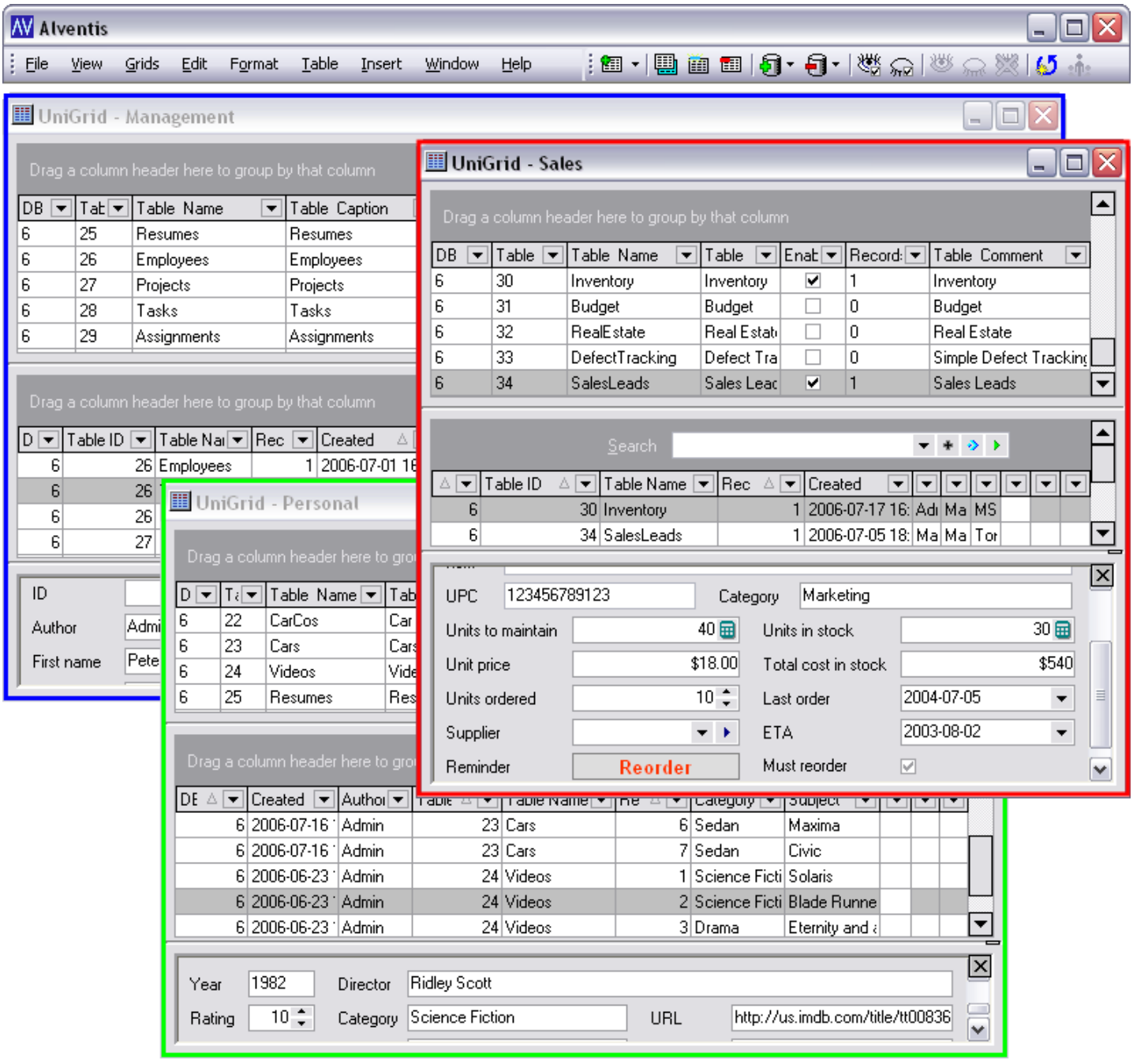
1. Tables Grid lists all accessible tables of the databases you have added to this UniGrid. You can add as many databases as you like to a UniGrid. Even databases located on different servers. Tables that you would like to participate in a search must be selected by ensuring their Enable checkbox is checked.
2. The Search Results pane has the Search Box where you enter a search expression, and a grid that will display the results of the search. More on this later.
3. The Preview pane will display the focused Search Results record.



Essentially, this is all you need to know to perform a simple search and find something.

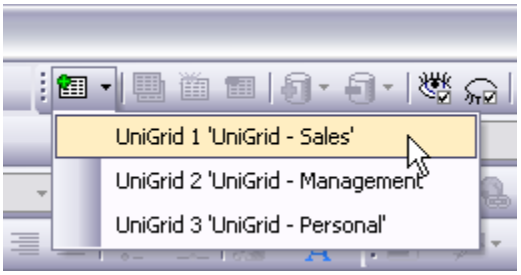
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

You can have as many UniGrid forms as you like. For instance, you may want to "dedicate" on UniGrid to a particular table or subset of tables, and another – to another set. Each UniGrid is independent. It can work with its own set of enabled tables from its own set of databases. It also maintains its own list of most recently used search expressions.



Since all of this might be of some value to you, a UniGrid maintains its settings even if you close it (to temporarily get it out of the way – or for any other reason). A closed UniGrid can be "brought back to life" by clicking the little dropdown arrow on the right of the Create/Reopen UniGrid button .

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



You'll see a list of all closed UniGrids that Alventis is aware of. Select the one you want – and it's back the way it was when you last closed it. Clicking on this same button (as opposed to opening the dropdown by clicking the little arrow) creates a brand-new blank UniGrid.

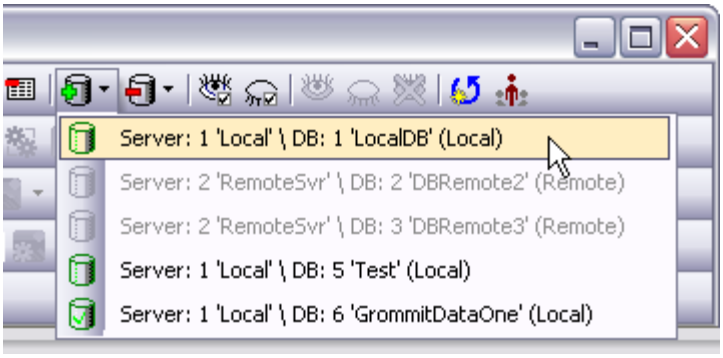
Clone UniGrid creates a new UniGrid which is an exact copy of the currently focused one.

Rename UniGrid allows you to change the Title of the current UniGrid to whatever may help you better identify it.

Kill UniGrid commits the focused UniGrid to oblivion. You'd do this when you want to permanently get rid of a UniGrid you never want to see again. This only removes the UniGrid form, it does not affect any data it may have been displaying, so this is not a very "destructive" operation even if it sounds menacing.

Before we can perform a search though, we must add at least one Database to the UniGrid.

The Add Database command drops down a list of all known Databases, of which the accessible ones are enabled.


Server: 1 'Local' \ DB: 1 'LocalDB' (Local)
Server: 2 'RemoteSvr' \ DB: 2 'DBRemote2' (Remote)
Server: 2 'RemoteSvr' \ DB: 3 'DBRemote3' (Remote)
Server: 1 'Local' \ DB: 5 'Test' (Local)
Server: 1 'Local' \ DB: 6 'GrommitDataOne' (Local)

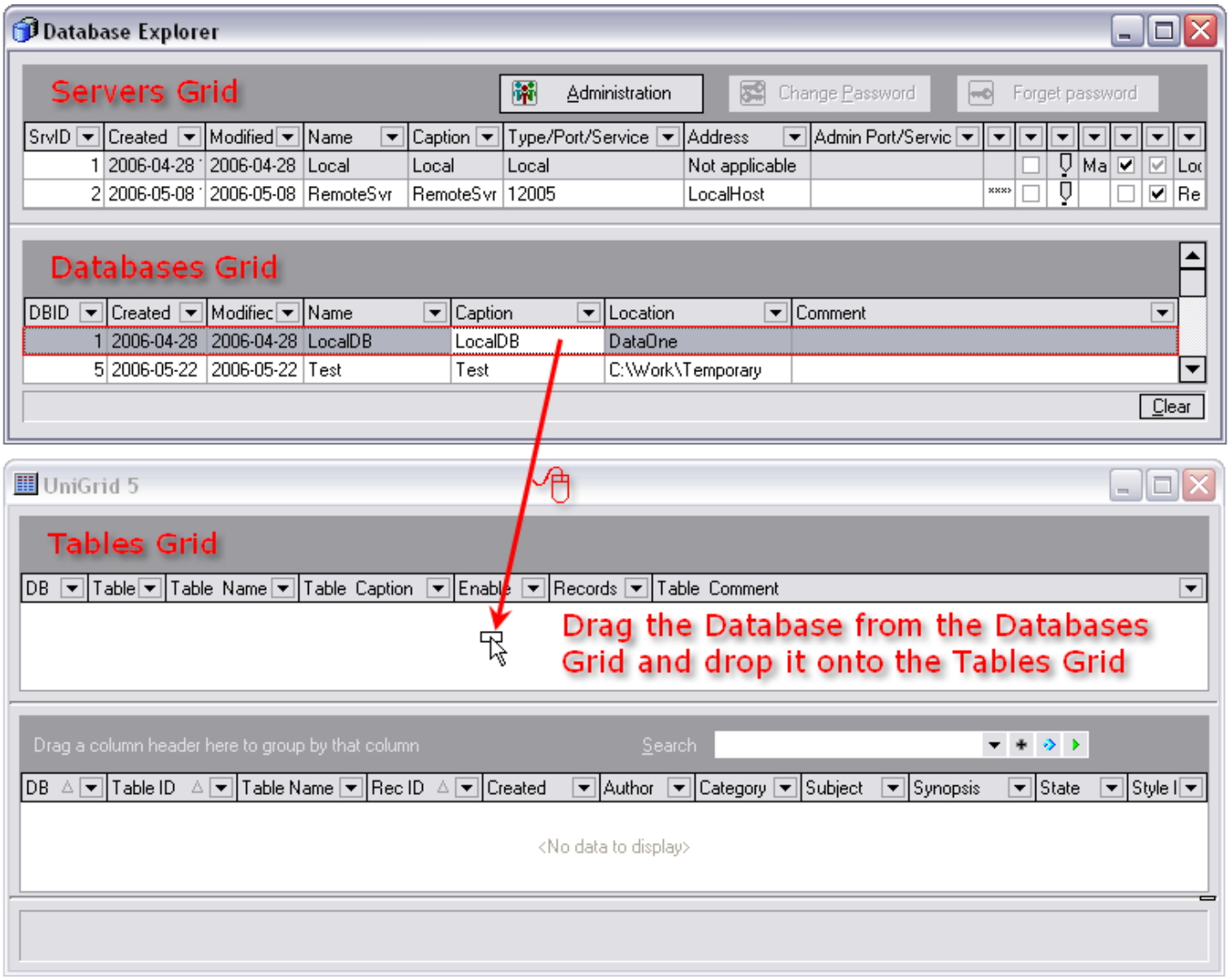
Picking a Databases from this list will add it (or re-add if it was already there) to the UniGrid. This means that all accessible Tables from that Database will now appear in the Tables Grid list. You can now enable the ones you want using the Enable checkbox.

The Remove Database command – as the name suggests – removes the Database you select in the dropdown menu from the UniGrid.
This does not affect the data of any Tables of that Database: it merely tells the focused UniGrid to no longer list any Tables from it. Nothing destructive here.

There's another way to add Databases to UniGrids. You simply drag the Database you want from the Databases grid in the Database Explorer form, and drop it onto the Tables Grid where you want to add

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

it.



Search Box.

We will talk much more about what exactly you should type in the Search Box in the Search Syntax chapter. But there's one thing we'd like to point out right here. Let's say you are looking for records mentioning "Caesar salad". In all likelihood, you won't even have to type the whole thing. The moment you have finished typing "Caesar", Alventis will find all records containing this word and you will see a list in the Search Results grid under the Search Box. Unless you have a table dedicated to Roman history, this list will be small enough to pick from it the record you are looking for. Quick and effortless.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

InstaSearch Mode. To find what you are looking for as quickly as possible, Alventis uses what is known as Full-Text Indexing. This means that whatever text you enter in a data table, Alventis organizes it in a special index the moment you save your work (i.e., when you post your changes). This way, this index called the Words Index is kept up-to-date at all times. As a result, when you perform a search looking for records containing the word Robert, Alventis doesn't need to go looking for it in every individual record of every table. All it has to do is find it in the Words Index, which will immediately supply Alventis with a list of records where this word is located. This procedure is usually so fast that Alventis performs a search for you "live", as you type.

This mode of operation is referred to as InstaSearch. Occasionally, you may find yourself working with tables that take a bit longer to search through even with Full-Text Indexing. This would typically happen with very large tables with tens of thousands of records or when the tables are located on a remote server to which you have a slow connection (e.g., via the Internet). InstaSearch should work fine even in such harsh conditions, but you may want to turn it off to avoid unnecessary intermediate searches while you are typing the search expression. You may do so by clicking the Toggle InstaSearch button on the righthand side of the Search Box. You can also use the Alt-Enter shortcut if the cursor is inside the Search Box. When in non-InstaSearch mode, you must initiate the search manually by either clicking the Perform Search button (rightmost in the Search Box) or simply hitting Enter on the keyboard.

InstaSearch mode is set on a per-UniGrid basis, so you can have it enabled in one UniGrid and disabled in another one.

The Search Box can remember the search expressions you have used in the past. In many ways, it's a regular MRU Combo Box ("MRU" stands for "Most Recently Used"). There's one important difference though, especially in the InstaSearch mode. If it remembered every single search performed, it would by now contain the following list of recent searches: "C", "Ca", "Cae", "Caes", "Caesa", "Caesar", since it *did* try to search for whatever you were typing. Not very useful, is it? This is why the Search Box only remembers the search expressions you explicitly tell it to remember. There are two ways to do so. You can click the Remember Current Search button on the right side of the Search Box.
You can also use a keyboard shortcut, which varies a bit depending on whether InstaSearch mode is On or Off. In InstaSearch mode, you only need to hit the Enter key while the cursor is in the Search Box (note that Ctrl-Enter works too). In non-InstaSearch mode, hitting Enter merely launches the search; to also have the combo box remember the expression, hit Ctrl-Enter.
To summarize this:

In InstaSearch mode:

Enter	Remembers the expression
Ctrl-Enter	Remembers the expression

In non-InstaSearch mode:

Enter	Launches the search
Ctrl-Enter	Launches the search and remembers the expression

Back to the Search Results though. Each located record is displayed on its own row. The dozen or so columns you see for the most part present to you some information about the record without going too far into specifics. For instance, you see what table and database this record is from (DB, Table ID, Table Name columns). This is followed by the System fields of the record (since all records have these in common): Rec ID, Created, Author, Subject, Category. So far, it's all quite simple and obvious. Less so with the remaining 3 columns.

The Synopsis column makes an attempt at presenting the most relevant information from the record –

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

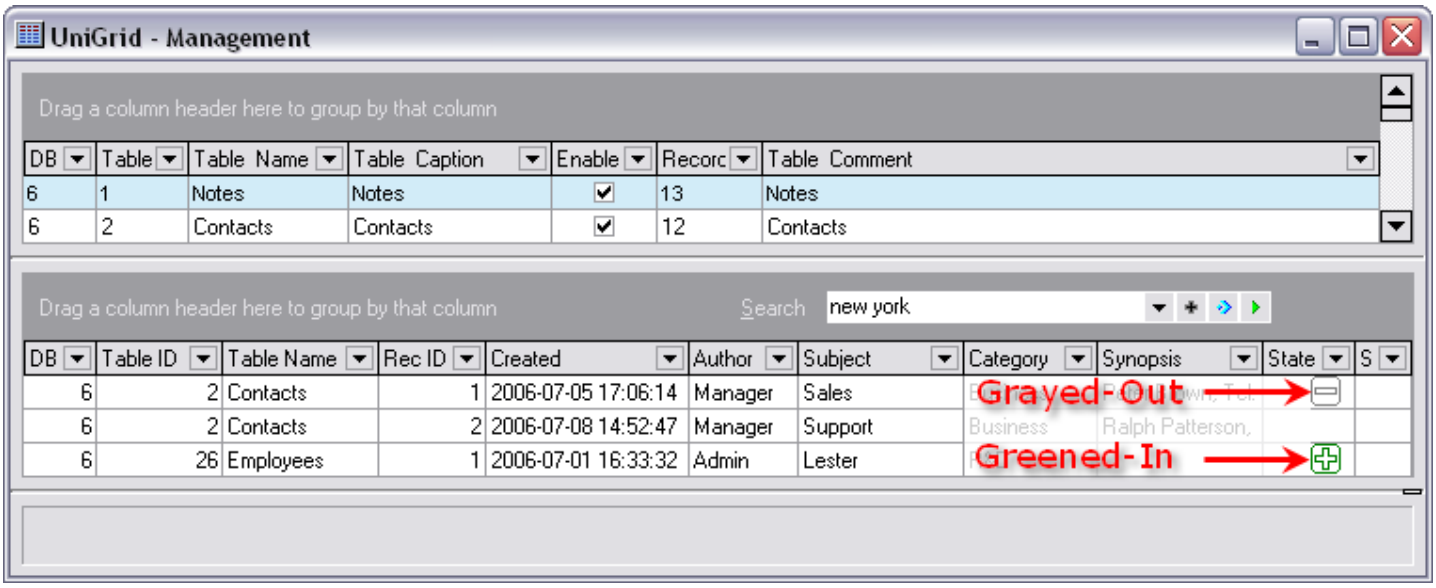
anything that didn't "fit" in the System fields/columns. What appears here may be different for each table. For instance, the Contacts records may benefit from displaying the First and Last Names they store. A Recipe may show the Dish Title field instead. Every table can display here whatever information from the record may be most helpful in identifying it at a glance. What specific field values appear here is configurable using the Synopsis Workbench which has a whole separate chapter dedicated to it.

The State column displays – you guessed it – the state of the record. The states we are talking about fall into two categories. The first of them is Hidden/Sticky. This subject is somewhat advanced, so you can learn about it in its own chapter.

The second state type is **Grayed-Out/Greened-In**. It's quite simple, really. Imagine you performed a search for "Richard" and you got 3 results that contain the following text:

- Richard's telephone number is 321-555-9876
- Richard III is a great tragedy and a very good film
- Al Pacino's "Looking for Richard" was nice too

Obviously, the word Richard is somewhere in all of them. So far, so good. Let's now open one of these records in its InfoView and edit it in such a way that "Richard" is no longer part of it. Perhaps we edited the whole text or decided to replace the reference to Richard by some other character. Whatever may be the situation, once we have posted our changes, the original Search Results are no longer correct: the record we have just modified no longer contains "Richard", so the Results grid should not be listing it as valid search result. On the other hand, maybe you don't care at that point in time about the "truthfulness" of the displayed Search Results: you just want it there so that you could gain access to it. This is why Alventis does not automatically remove the no-longer-matching result from the list. However, just leaving it there could be confusing if you ever forgot you edited it. So Alventis simply marks the record that no longer belongs in the Search Results due to your own modifications with a gray minus sign in the State column, which is what we term "Grayed-Out".



Conversely, you may edit some other record that was not mentioning Richard. If after your modifications the record starts to match an existing search, such a record is automatically added to the previously listed Search Results. To better distinguish it, it is added with a green plus sign and is said to be "Greened-In".
That's all there is to it. Grayed-Out/Greened-In states simply give you visual queues to what is

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

happening to existing Search Results as a consequence of your modifications of records. Naturally, these states are cleared the moment you perform a new search.

The Grayed-Out/Greened-In updates of Search Results can be disabled using the On-Edit Search Updates toggle button. The re-evaluation of searches may take a few seconds, especially if you have multiple open UniGrids. You may therefore want to temporarily disable such automatic updates if all you are interested in doing is quickly updating multiple records.

Finally, the Search Results grid is adorned by the Style Name column. Stylizing of records is discussed in a separate chapter, so we won't deal with it at the moment. Suffice it to say that records may have Styles attached to them, and this is where you see the name of the record's Style (if any).

Once you have obtained some search results, you may want to open some of them in an InfoView form. InfoViews are described in detail in the next chapter, but we'll quickly tell you here how you open them.

- You can double-click the Search Results record
- You can hit Ctrl-Enter on the keyboard to open the focused Search Results record
- You can open a Search Results record in any existing InfoView that you can see on your screen by dragging the record and dropping it on either the Titlebar or the background of the DataNav bar at the top of the InfoView. That target InfoView doesn't have to belong to the same table as the record you are trying to open. It does have to be in "saved" state though, i.e., it must not be in the middle of an editing operation.
- You can open an InfoView to create a new record in its table by double-clicking the corresponding table in the TableGrid.
- You can invoke the Open Record Wizard from the File menu (or hit the Ctrl-O keyboard shortcut).
- Finally, you can create a new record using the New Record Wizard accessible from the File menu or via the Ctrl-N shortcut.

By default, when you open a record in an InfoView, Alventis does it the following way. If this record is already open in some existing InfoView, that InfoView is simply brought to the front in case you lost it under other forms. If this record is not open yet, it is opened in a new InfoView. This is where you have some choice though. Alventis Settings are described elsewhere, but if the Records Open Mode is set to "Open records in an existing form of the same table", this is precisely what will happen: Alventis will attempt to locate an existing open InfoView of the same table, and if it finds one, that's where it will open the record. If it doesn't find a suitable InfoView it will open a new one of course. You can temporarily "invert" whatever Records Open Mode is currently in effect by opening a record while holding down Shift on the keyboard: for example, if the application Settings are currently set to "Open records in an existing form of the same table", opening a record while holding down Shift, will open it in a new form.

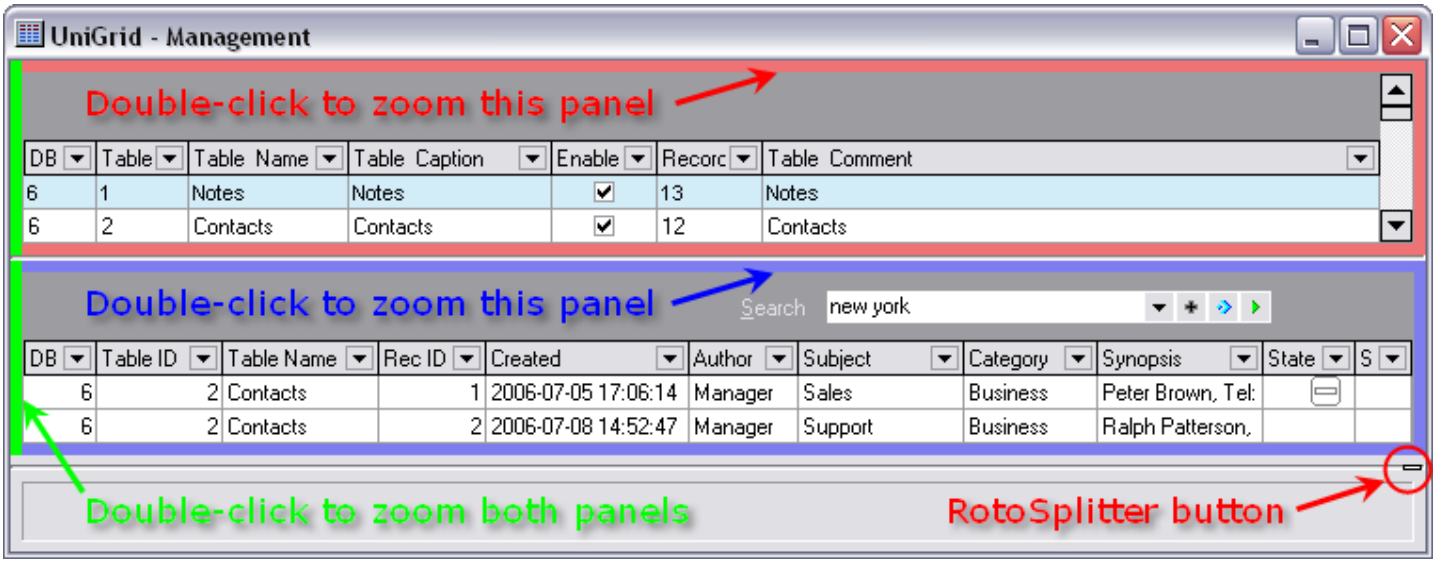
The Using Grids chapter describes in some detail what you can do with a grid, and all of it applies to the Search Results grid.

At the long last, we get to the **Preview pane**. The Preview pane affords you a quick glance at the Search Results records – one record at a time. Just highlight any record and it will get displayed in the Preview pane. What you see there is actually a whole InfoView displaying the record in question – all squeezed into the Preview. Depending on the size of the Preview pane and on the contents and layout of the InfoView, it may or may not fit neatly. The goal is to give you the ability to quickly inspect the found records in more detail than can fit in the Search Results grid. To really work with the record you're interested in, you would normally open it in its own InfoView form – completely independently from the UniGrid that found it. This is explained in more detail in the next chapter.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Some InfoViews may be sufficiently simple and compact to nicely fit in the Preview. Or perhaps your monitor is so huge that you could resize the Preview pane so that whatever you want fits there. In any case, while the Preview was never intended as your main way of accessing the data, you can do so if you prefer. For example, you can edit existing data. The moment you modify the displayed record, two small buttons appear in the top/right corner of the Preview: Post and Cancel. They are similar to what you have all the time on the DataNav toolbar in a "real" standalone InfoView. So, you can edit data directly in the Preview. What you cannot achieve here is: deleting records, creating new records, and navigating between records of the same table. All of this can only be done in a standalone InfoView.

The UniGrid form is of course resizable, as are the 3 panes it is comprised of. You can drag the splitters between the panes wherever you want. You can zoom-in to one or both of the grid panes. PanelZoom is described in a separate chapter, but en bref: double-click in the narrow area between the grid of interest and the margin of the corresponding pane: the pane you double-clicked temporarily fills the entire UniGrid. Double-click it again to go back to the normal multi-pane view. With UniGrids, there's a little trick to PanelZoom though. Double-clicking just to the left of either grid makes you zoom-in on both grid panes at the same time (effectively just hiding the Preview). Finally, you can rotate the splitter that separates the grids from the Preview. As described in more detail in that same User Interface chapter, either click the little button at the right/bottom edge of the splitter or Shift-double-click anywhere on the splitter.



There's one last feature we'd like to mention here even though it is not related to UniGrids per se.

The Delete Data Table button deletes the table selected in the Tables Grid of a UniGrid. The table will be permanently deleted from wherever it resides on disk, so exercise caution!

2.2 InfoViews

InfoView forms constitute your main means of accessing data records. Each InfoView corresponds to a particular table that it "represents".

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

An InfoView may present to you more than a single table if the tables are relationally linked. This advanced topic is described in a separate chapter though, so for the remainder of the present discussion we will be assuming we are dealing with a relatively simple InfoView based on a single table.

All InfoViews have one thing in common: they have a DataNav toolbar at the top. This toolbar allows you to navigate between records of the table this InfoView belongs to, as well as manipulate the currently displayed record. The buttons, their functions, and their keyboard shortcuts (where available) are listed below:

First	Ctrl-Alt-PageUp	Displays the first record in the table
Prior	Ctrl-PageUp	Moves you to the preceding record
Next	Ctrl-PageDown	Goes to the next record
Last	Ctrl-Alt-PageDown	Goes to the last record
Insert	Alt-Insert	Creates a new record
Delete	Alt-Delete	Deletes the current record
Post	Ctrl-S	Posts your editing changes (i.e., saves the record)
Cancel	<None>	Abandons your changes
Switch InfoView	<None>	Switches to another InfoView if there are more than one or refreshes the current one

Some notes are in order. New records are always appended at the end of the table. They automatically get the next available ID value (which is why you can't edit the ID value by hand). They also get their Author, and Creation and Modification Dates set automatically.

If you edit a record and navigate to another record without explicitly posting your changes, they are posted for you automatically. Same thing happens if you attempt to create a new record while in edit mode: editing changes are posted, and then the new record is created. Incidentally, this makes it a bit quicker to create one record after another: create, fill-in the data, create next one, etc.

Buttons may be enabled or disabled depending on where you are in the table and what rights you have with respect to the table or the current record. Rights only play a role if you are accessing a secured Alventis server, so we won't get into them here.

You may encounter two types of InfoViews.

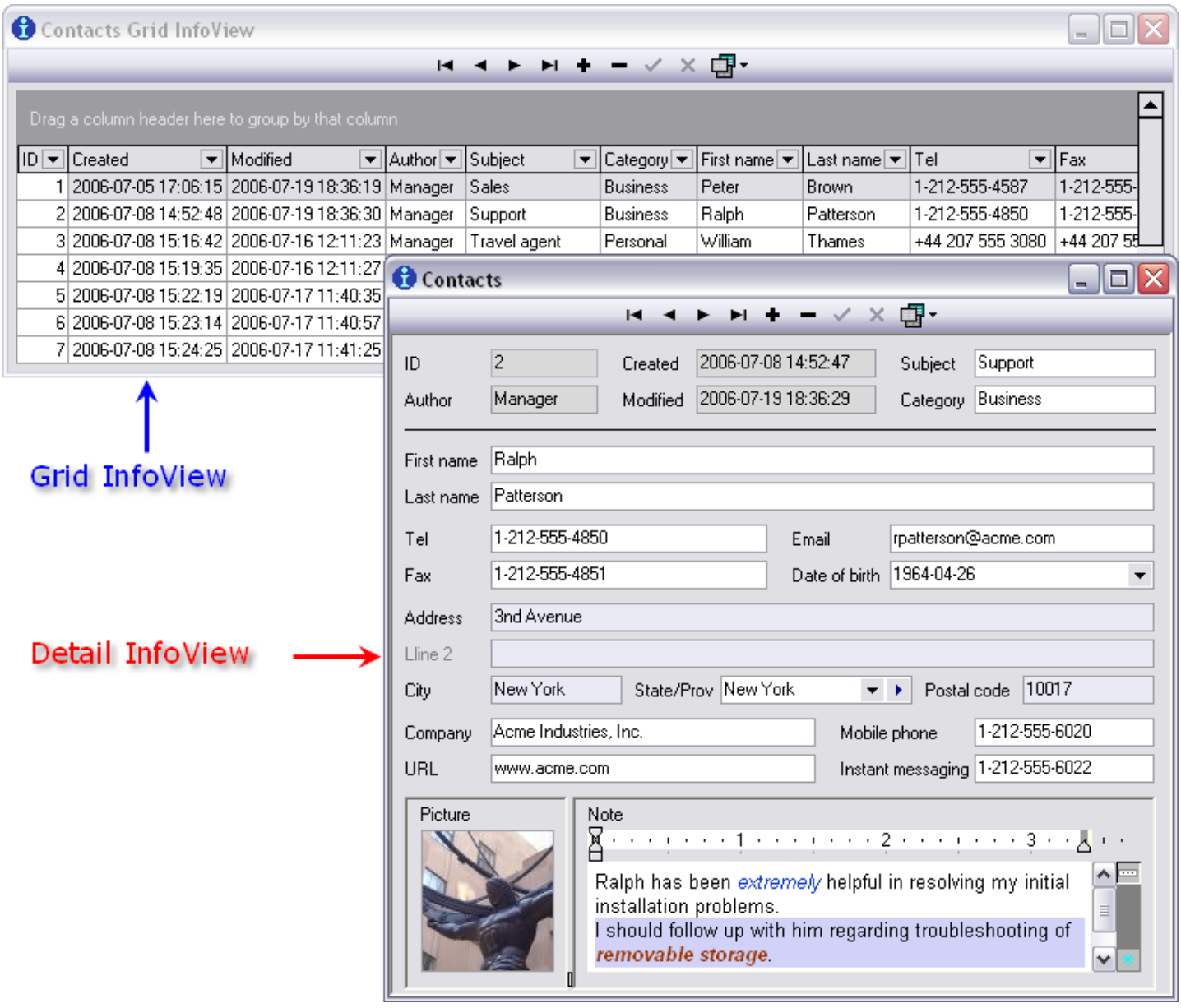
Detail InfoView forms will have fields represented by individual Text Boxes, Memo Editors, Combo Boxes, Radio Groups, Image Boxes, and so on.

Grid InfoView forms will represent their table using a grid. Grids allow one to see several records at the same time. However, the concept of current record still applies. For instance, if you edit a record in a grid and attempt to focus another record, your changes will be posted. Grids offer some features that are quite unique and useful. You can learn everything about them in a separate chapter. On the other hand, they have some limitations. Since they are displaying an entire record on a single line, space is more limited and things may become quite crammed. Even more importantly, grids cannot display

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Memos. Not all is lost though.

Which brings us to Mixed InfoView forms. Such forms may have a grid that represents some fields of the table. Whatever doesn't fit in the grid may be represented by our good old Text Boxes, Memo Editors, and so on – all scattered (neatly of course!) around the grid. The grid is then still displaying all the records of the table at once. The other controls though are only displaying data from the current record. Including the Memo that you couldn't see directly in the grid.


Contacts Grid InfoView
Drag a column header here to group by that column
ID Created Modified Author Subject Category First name Last name Tel Fax
1 2006-07-05 17:06:15 2006-07-19 18:36:19 Manager Sales Business Peter Brown 1-212-555-4587 1-212-555-
2 2006-07-08 14:52:48 2006-07-19 18:36:30 Manager Support Business Ralph Patterson 1-212-555-4850 1-212-555-
3 2006-07-08 15:16:42 2006-07-16 12:11:23 Manager Travel agent Personal William Thames +44 207 555 3080 +44 207 55
4 2006-07-08 15:19:35 2006-07-16 12:11:27
5 2006-07-08 15:22:19 2006-07-17 11:40:35
6 2006-07-08 15:23:14 2006-07-17 11:40:57
7 2006-07-08 15:24:25 2006-07-17 11:41:25
Grid InfoView
Detail InfoView
Contacts
ID 2
Created 2006-07-08 14:52:47
Subject Support
Author Manager
Modified 2006-07-19 18:36:29
Category Business
First name Ralph
Last name Patterson
Tel 1-212-555-4850
Email rpatterson@acme.com
Fax 1-212-555-4851
Date of birth 1964-04-26
Address 3nd Avenue
Lline 2
City New York
State/Prov New York
Postal code 10017
Company Acme Industries, Inc.
Mobile phone 1-212-555-6020
URL www.acme.com
Instant messaging 1-212-555-6022
Picture
Note
Ralph has been extremely helpful in resolving my initial installation problems.
I should follow up with him regarding troubleshooting of removable storage.

A table can have more than a single InfoView representing it. You may, for example, want to have a Detail InfoView and a separate Grid InfoView: depending on what you are doing, each has its advantages. Another table may have two Detail InfoViews with different layouts that present the same data differently. Whatever the case may be, you can switch between InfoViews at any time using the rightmost button on the DataNav bar . Clicking it displays a menu of accessible InfoViews. The

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

current one is shown with a blue icon next to it. If you select the same InfoView you are already using, it will get re-opened thus refreshing the current record (so you will see changes you or somebody else may have made to this record while you had it open). If you select another InfoView, your current record will be displayed in whatever other form you have selected. In all cases, if you do the InfoView switch in the middle of editing a record, your changes will be lost, so if this is not what you want, post them prior to switching.

Whatever InfoView form you select, your choice is remembered by Alventis, so this InfoView will be used next time you want to open a record from this table – be it in a standalone InfoView or in the Preview pane. It should be noted that you may have more than one copy of the same InfoView open at the same time, displaying the same or different records from the same table. One may be confined to the Preview pane, others may be floating around. When you switch to another InfoView form, this will only immediately affect the InfoView from which you initiated the switch: the other InfoViews will remain as they were until you close and re-open them (or switch them too – except the Preview that can't initiate a switch). You can sometimes use this to your advantage: perhaps by using one InfoView in the Preview form while working in a different standalone InfoView of the same table.

Other than what has been described above, InfoViews can come in all shapes and sizes (as long as they are rectangular and fit within Alventis' Workspace, to paraphrase Henry Ford). Basically, the contents and layout of an InfoView is very much up to its creator, i.e., any user armed with a copy of Alventis Professional.

2.3 Alventis and Designer Interface Overview

Alventis and Designer use an innovative windowing interface allowing you to work in multiple concurrently open forms. The concept should be familiar to you from numerous other applications using the standard Microsoft Windows Multiple Document Interface (MDI). Alventis and Designer are similar in many ways, but differ in some important details. Since both Alventis and Designer use the same exact interface, and this entire chapter applies equally to both, we will be only mentioning Alventis for the rest of this discussion for the sake of brevity.

A large part of Alventis' interface follows the "standard" quite closely:

- You can open as many forms as you like. Note that we call them "forms" but many applications refer to them as "windows".
- You can arrange forms within the Alventis Workspace (the empty area of the main application form) whichever way you like.
- All forms are resizable. Note that we are talking here only about forms that can reside in the Alventis Workspace concurrently with other similar forms. This excludes dialog boxes, for example.
- You can minimize forms. They appear as little rectangles at the bottom of the Workspace.
- You can maximize forms to make them as large as the Workspace.
- You can activate the desired form by picking it from the Form List dropdown menu . You can also switch between the current and the previously-active form using the Alt-F6 shortcut.

As you can see, there's nothing unusual about the most essential aspects of Alventis' interface. There are only two new and unusual elements:

- Forms are Workspace-bound. Not only do they appear within the Alventis Workspace, but they cannot extend beyond its boundaries. You simply can't move or resize a form to make it not fit. Resizing the Workspace moves and/or resizes forms within it if necessary. A very nice consequence of this aspect of Alventis' interface is that it becomes remarkably easy to place a form, say, in the

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

corner of the Workspace. You no longer have to carefully move it in the desired position only to overshoot by a pixel or two and get the dreaded Workspace scrollbars. In Alventis, you just move it as far as it will go: because it won't go any further than the corner or side of the Workspace.
- Maximize is per-form. When you maximize a form, only this form is maximized. The rest of them remain normal, so you can have a sizable form on top of a maximized one if this is convenient. You are free to maximize multiple forms, of course.

Interface Persistence. Alventis attempts to preserve the last known state of everything within it and restore it whenever you re-open the relevant form. Simply put, you can close Alventis that has two dozen open forms, re-open it the next day, and find everything in place as if you haven't closed it at all. The following is a partial list of the most important elements Alventis preserves for you:
- Form state and position
- Menu and bars setup
- InstaButton's individual settings
- Splitter positions
- Grid customizations
- MRU (Most-Recently Used) combo box's lists

You may occasionally want to defeat the automatic saving or loading of these Persistence states. Both saving and loading do not occur if the Shift key on the keyboard is pressed. Pressing and holding it starting with the moment the Splash Screen appears, for example, will launch Alventis "clean" with nothing re-opened and re-loaded. Same goes for InfoView forms: Shift-opening them will prevent their Persistence loading. You may not be able to successfully exercise "fine control" over what you do and don't want to save/load, i.e., it is not possible for you to launch Alventis re-loading its menu/bar setup but not MRU lists of combo boxes. Nevertheless, you can certainly use this feature to, say, open an InfoView, mess-up its layout with its Splitters and Grids, and Shift-close the InfoView without saving any of your unwanted changes for posterity.

Whenever you launch Alventis with the Shift key pressed, the first thing it does is offer you to clear all Persistence information completely. You can accept or reject this offer based on the circumstances.

Alventis has some menus and toolbars. On these menus and toolbars, you will find buttons, combo boxes, and other interface elements that together expose almost all the features available to you. By default, both menus and toolbars contain almost all available items. Together, they contain all of them. The items not duplicated on the menus are simply more appropriate for toolbars: Bold/Italic/Underline/Strikeout buttons, as well as Font Name, Size, Style Picker, and MemoSearch combo boxes. Toolbars and menus are essentially equivalent in the sense that they give you access to whatever items they happen to contain. Clicking a button on a toolbar is therefore no different from clicking the same button on a menu. Which is why we won't be making any explicit distinctions about bars and menus when we talk about clicking buttons or using other similar items. Whether you prefer to use menus or toolbars is a matter of your personal taste. We tend to favor toolbars since they offer a more direct access to the buttons they contain. In particular, toolbars are more convenient when dealing with numerous InstaButtons. That said, you can have multiple InstaButtons on menus too, so it's completely up to you.
You can fully customize both menus and toolbars to your liking. See the Customizing Toolbars and Menus chapter for more information.

Alventis supports a variety of common data controls, such as Edit Boxes, Combo Boxes, Radio Button Groups, and so on. These should be familiar to you from other software, so we won't be describing these

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

in any detail.

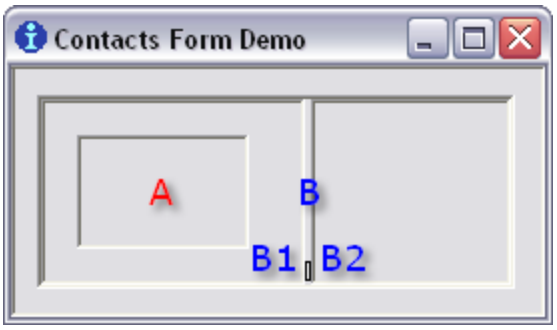
You may be less familiar with a few other items, so we'll give you brief tips on their use.

- SpinEdit Boxes allow you to enter numeric values. Some are limited to integers, others allow floating point numbers. You can enter values directly as you would in a regular Edit Box. You can also "spin" the value incrementing or decrementing it using the little up/down buttons. You can do the same using the keyboard by hitting the Up/Down keys. You can also use PageUp/PageDown to increment/decrement the value in larger steps. For example, Up may increment by 1, while PageUp would increment by 4.
- Image Boxes can store pictures. All operations available to you are accessible via the context menu. To open it, right-click the Image Box.
- Calculator, Date, and Time Boxes work on the same principle: you can enter values directly as you would in an Edit Box or you can use the drop-down dialogs that allow you to calculate or pick the desired value.

RotoSplitters. These are the fairly common draggable dividers you should be familiar with from other software. The only unusual thing about splitters in Alventis is that they can be rotated. Look for a little "button" at the right or bottom edge. Splitters that have this button are RotoSplitters. You can rotate them, i.e., toggle their orientation between horizontal and vertical, by clicking the little button or by Shift-double-clicking anywhere on the splitter. A few splitters cannot be rotated: usually, such splitters have an obvious preferable orientation, so changing it simply wouldn't make much sense.

PanelZoom. Alventis allows you to "zoom-in" to a particular Panel (or area of a form). This amounts to "maximizing" that Panel to temporarily fill the entire form – much as you would maximize a form to fill the entire Workspace of the application. There are two ways to zoom a Panel. You can double-click the desired Panel. Note that you can double-click not only the Panel per se, but any Label, Shape or static (cosmetic) Image on it – anything on the Panel that normally does not react to clicks. The other way is to invoke the PanelZoom command from a toolbar or a menu. This will zoom the Panel closest to the currently focused item on the form. This seemingly simple statement hides a lot of complexity, so let's try to see what it is really trying to tell you.

Alventis always tries to zoom the closest enclosing Panel that supports zooming. And not all Panels do. Time for an example.



Let's imagine that the designer of the form depicted above has set the following Panels to be zoomable: A, B1, and B2. This case is fairly simple: you can zoom the entire Panel A by double-clicking it. You can also zoom each of the split sub-panels B1 and B2.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

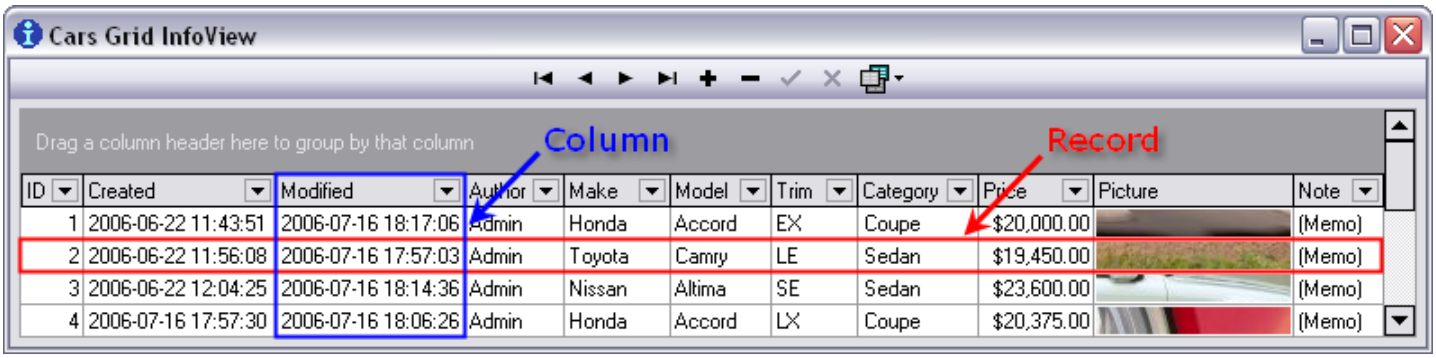
Let's now imagine that the designer has specified only Panel B to be zoomable. Attempts to zoom either sub-panel B1 or B2 will no longer zoom the corresponding sub-panel, but rather the entire Split Panel B. Let's say we double-clicked B1. This sub-panel is not zoomable though, so Alventis will start looking for the next enclosing Panel. In the above example, it won't have to look very far: the very next Panel is the entire Split Panel B, which is zoomable, so this is what gets zoomed. Similarly, attempts to zoom Panel A will result in Alventis going through Panels in the order A → B1 → B, and zooming B again.

What Panels are or are not zoomable is up to the designer of the form. As far as InfoView forms are concerned, this is set on a per-Panel basis in Alventis Designer.

2.4 Using Grids

Grids are perhaps the most important user-interface element in both Alventis and Designer. It is also one that is both exceptionally powerful and rather complex, so we will take a close look at Grids in this topic.

A Grid has columns and records that are sometimes also called rows.



In its most basic shape depicted above, you can scroll around to view all available records and their columns. You can also edit their values using the Grid cells. In most cases, you will first focus the desired cell by either clicking it with the mouse or by navigating to it with the keyboard's arrow/cursor keys. The record is now highlighted, and the focused cell within it is inverse-highlighted, which is a euphemism for saying "doesn't look highlighted".
To edit the cell's contents, you have several choices:

- Start typing. The moment you start typing, you find yourself "inside" the cell, entering the text you already started typing. Old text, if any, gets replaced.
- Hit Enter. You find yourself in the cell with its old text selected. This gives you an opportunity to edit the existing text.
- Click the focused cell with the mouse. Works just like Enter above.

There are a few Grids in Alventis and Designer that behave slightly differently, at least in some of their columns. The difference can be that as soon as you click a cell with the mouse, you find yourself right away inside the cell (as if you had already clicked the cell once, and then the second time). Other than that, everything else works the same.

Once you have entered the new text in the cell, you have a few ways to get out:

- Hit Enter. This may do one of two things, depending on the Grid: it will typically just "close" the cell and you will find yourself as you were before you entered into it: the cell is focused but not active.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Some Grids will automatically place you in the next column's cell as a service to you.

- Hit Escape. Your modifications, if any, are abandoned and you're out of the cell.
- Hit Tab. You are out of the cell and the next cell gets focused, but not activated, i.e., you're not "inside" it.
- Click somewhere with the mouse. This is terribly boring, but it will accept your changes and focus the clicked cell.

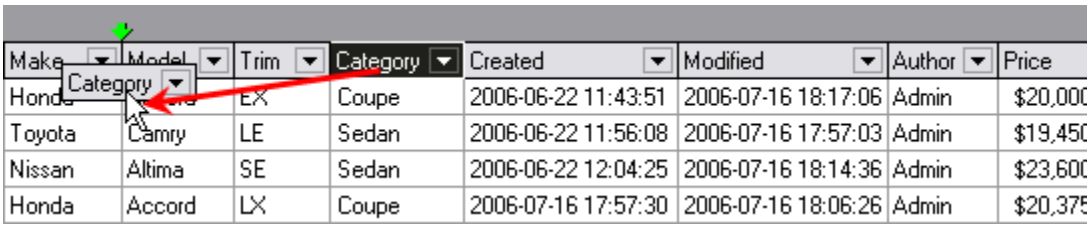
Creating Records. You can insert a new record in a grid by focusing the grid (or any cell within it) and hitting the Insert key on the keyboard. Some grids do not allow creation of new records, in which case nothing will happen.

Deleting Records. You can delete the focused record by hitting Delete or Ctrl-Delete on the keyboard. Once again, some grids may not allow you to delete their records.

Saving Records. Once you have modified one or more cell values for a record, you must post your changes (i.e., save the record). To do so, you must simply focus any other record in the grid: just moving up or down by one record is sufficient. If the modified record is the only record in the grid, you can achieve the same effect by "pretending" to insert a new record: hit the Insert key to create a new record. You find yourself in the new empty record, and the previous modified record has been posted since it is no longer focused. You can now hit Escape to cancel the insert operation (unless inserting just happens to be exactly what you want to do at this time). In an InfoView grid, you can also post the changes using the Post button on the InfoView's toolbar.

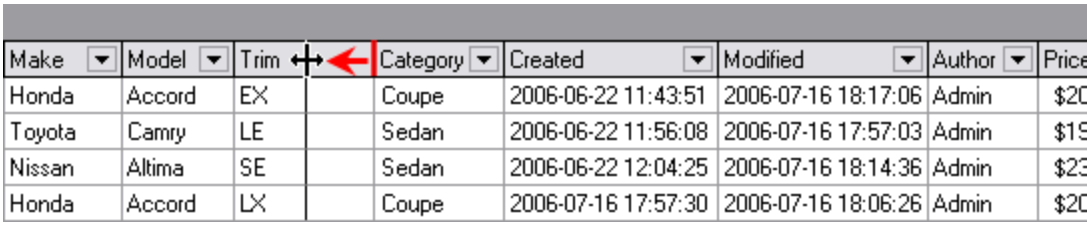
Moving around in the grid is quite intuitive. If you are currently in an active cell, you may want to get out of it though by clicking Enter, Escape, or Tab as described above. Once the cell is merely focused, you can focus and cycle through cells and records using the Tab key. You can also move in the desired direction using the cursor/arrow keys, and all the other usual document-navigation keystrokes: Home, End, Ctrl-Home, Ctrl-End, PageUp, PageDown, Ctrl-PageUp, Ctrl-PageDown.

Moving Columns. In most Grids, you can re-order columns. Simply drag the column by its header (caption rectangle) from its current position to a new one between other columns.



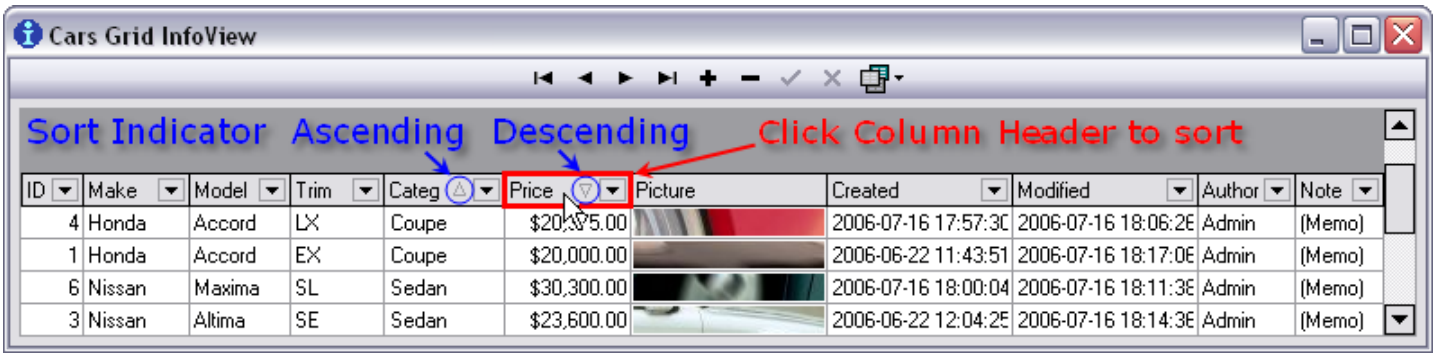
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Sizing Columns. You can make columns as narrow or as wide as you wish. Just drag the divider line between column headers where you want. Double-clicking the divider line sets the width of the column just to the left automatically according to the contents of the column.



Make	Model	Trim	Category	Created	Modified	Author	Price
Honda	Accord	EX	Coupe	2006-06-22 11:43:51	2006-07-16 18:17:06	Admin	$2C
Toyota	Camry	LE	Sedan	2006-06-22 11:56:08	2006-07-16 17:57:03	Admin	$1S
Nissan	Altima	SE	Sedan	2006-06-22 12:04:25	2006-07-16 18:14:36	Admin	$23
Honda	Accord	LX	Coupe	2006-07-16 17:57:30	2006-07-16 18:06:26	Admin	$2C

Sorting. Most Grids allow sorting. Sort by the desired column's values by clicking the column's header with the mouse. To toggle between Ascending and Descending sort order, just click the column header again. You will notice a little sorting arrow appear in the header of the column by which the Grid is sorted.

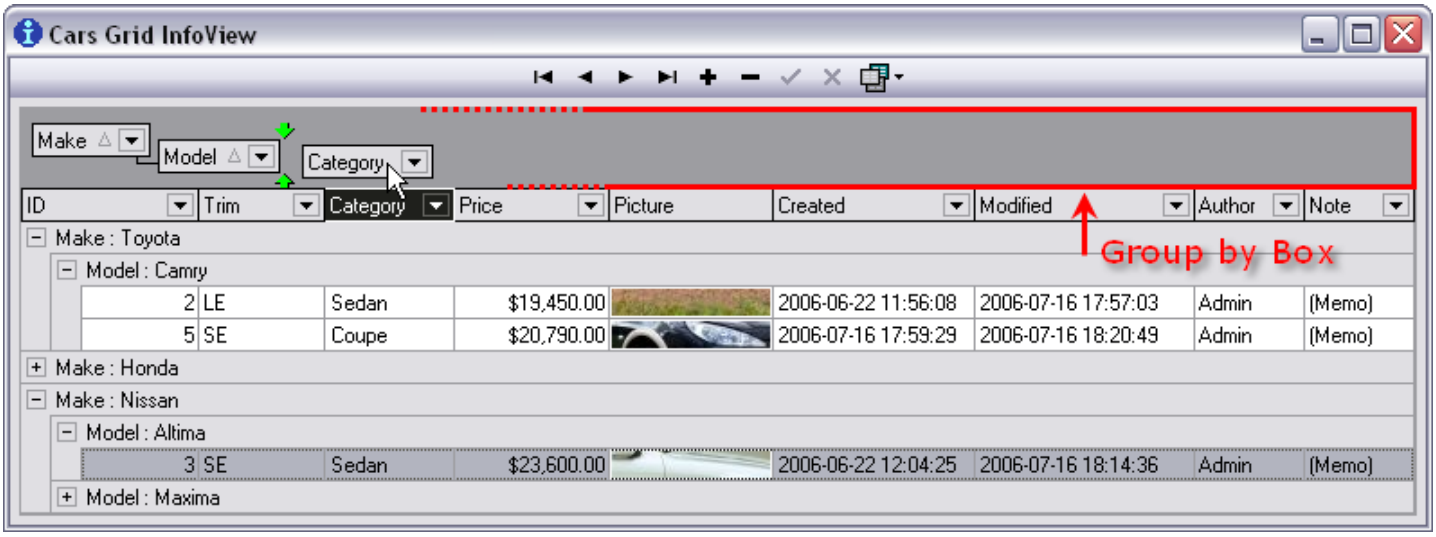


ID	Make	Model	Trim	Categ	Price	Picture	Created	Modified	Author	Note
4	Honda	Accord	LX	Coupe	$20,[illegible]5.00		2006-07-16 17:57:3C	2006-07-16 18:06:2E	Admin	(Memo)
1	Honda	Accord	EX	Coupe	$20,000.00		2006-06-22 11:43:51	2006-07-16 18:17:0E	Admin	(Memo)
6	Nissan	Maxima	SL	Sedan	$30,300.00		2006-07-16 18:00:04	2006-07-16 18:11:3E	Admin	(Memo)
3	Nissan	Altima	SE	Sedan	$23,600.00		2006-06-22 12:04:2E	2006-07-16 18:14:3E	Admin	(Memo)

You can remove sorting from a column by Ctrl-clicking its header.
You can also sort by more than one column. Shift-clicking a column header preserves whatever sorting may exist in any other columns, essentially adding that column to the list of sorting ones.
Sorting may not be supported either by the Grid or by a specific column. You will notice this... well, when it doesn't work.

Grouping. Grouping is an exceptionally powerful feature that arranges grid records by common column values.
You may be familiar with the concept of Grouping from other software. If not, playing with it for 2 minutes should be enough to understand how it works and what it does, so we'll offer just one simplistic example. Imagine a grid full of car records. Each record will list such things as Make and Model – each in its own column. If you group the grid by the Make column, that column no longer appears in the grid itself. Instead, each unique Make value now appears in its own grouping row. Under each of these grouping rows, will be listed records that correspond to that Make.

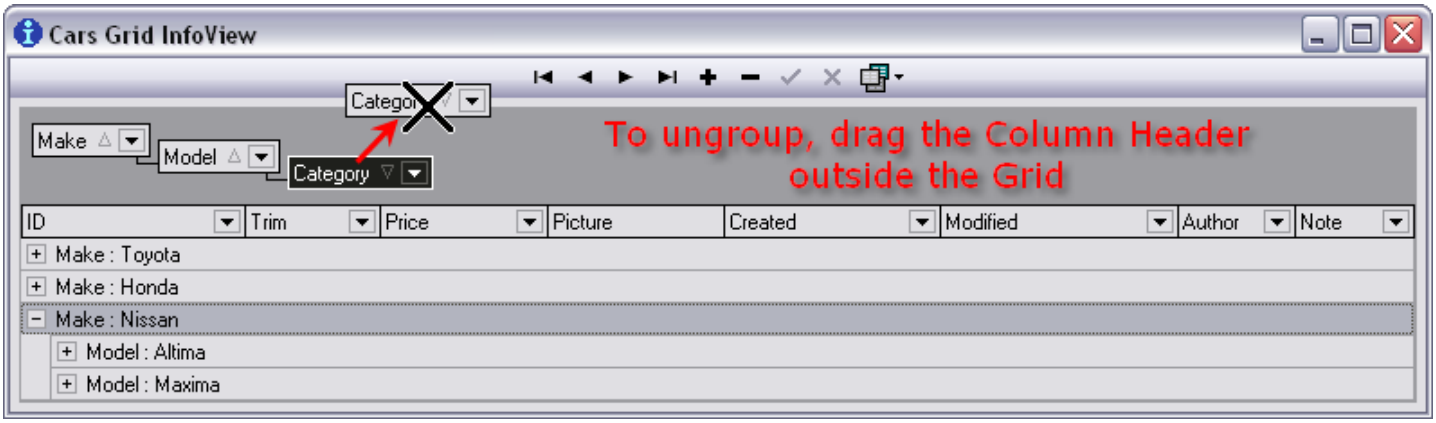
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



To group the Grid by a particular column, drag that column's header into the Group by Box at the top of the Grid. Grids that don't display this box do not support grouping.

You can group by more than one column at a time. Just repeat grouping for as many columns as you like.
You can group in a specific order and rearrange that order directly in the Group by Box by dragging column headers where you want them to appear in that multi-level grouping hierarchy.

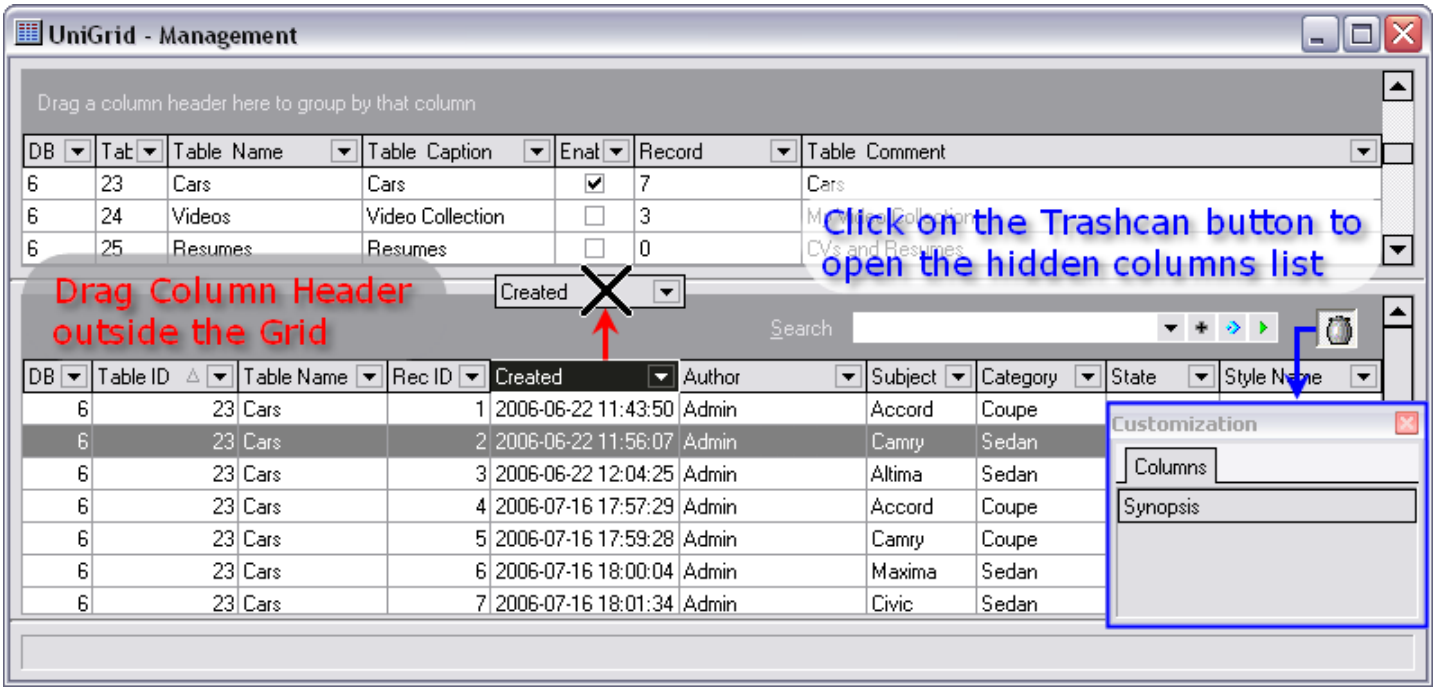
Ungrouping. You can ungroup by dragging the column header from the Group by Box back into the Grid's header row. You can also "put it back" by dragging it from the Group by Box and dropping it outside the Grid.



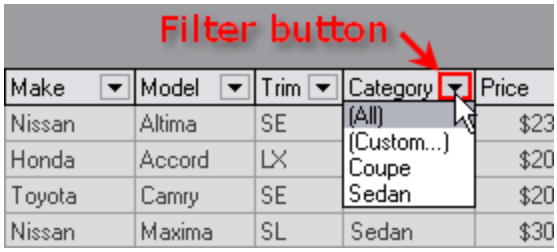
Grouping implies sorting, so if you group the grid by a column, it gets automatically sorted by that column. You can change the sort direction by clicking the column header as usual.
You can expand or collapse individual grouping rows. You can click the little Plus button with the mouse, double-click anywhere on the row (except in the Queries List grid where this action opens the double-clicked Query), or hit the '+' key on the keyboard (the one in the numeric keypad area on the far right). To collapse an open group, click the little Minus button with the mouse, double-click the row, or hit the '-' key (again, in the numeric keypad area). The numeric keypad '*' key will expand a group and all its sub-groups in case of multi-level grouping.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

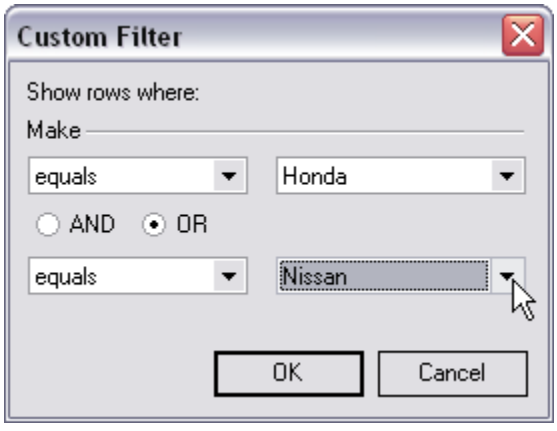
Column Hiding. Only the UniGrid Search Results Grid supports column hiding. You can hide columns you don't wish to see by dragging them outside the Grid and dropping them into "nowhere". The column vanishes from the Grid. To unhide the column, click the Trashcan button that is now shown in the right-hand corner of the Group by Box. This displays a pop-up Customization list of hidden columns. You can drag them back into the Grid's header row or Group by Box.



Filtering.
Many Grids allow you to display a subset of available records by specifying certain Filter Criteria. The simplest kind of criterion is by a column's value. Click the Filter button in the right corner of the column's header.
You are presented with a menu of the following choices:
(All) means "show all records", i.e., no filtering by this column



(Custom...) will open a Custom Filter dialog box for the current column. It allows you to specify a filtering expression with a few logical operators.

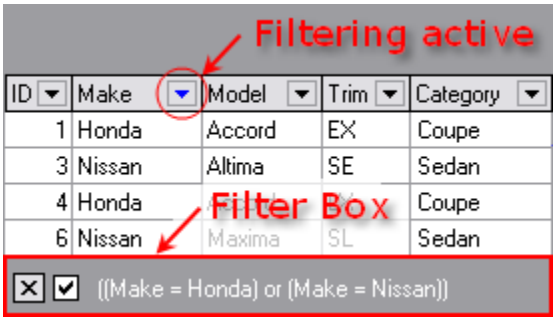


Values – the rest of the menu displays a potentially long list of values found in this column. Selecting a particular value will simply set the filter expression to "Column value must equal <whatever you chose>".

You can filter in the above fashion by more than one column.

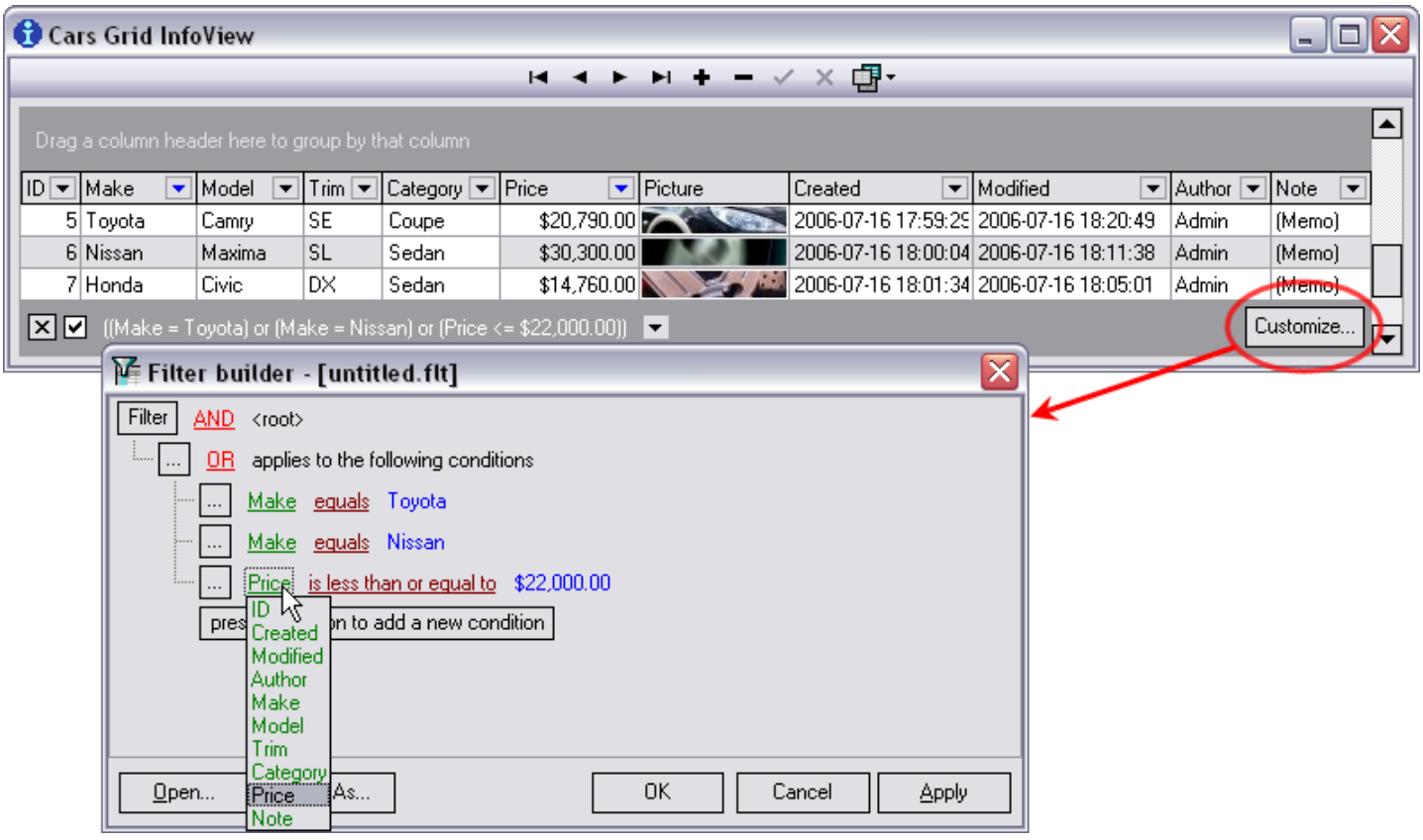
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Whenever filtering is in effect, a Filter Box appears at the bottom of the Grid.



It textually displays the current filter expression. You can remove filtering completely by clicking the gray Close Filter button on the left. The checkbox right next to it will temporarily disable the filter but will not remove it. A little "down" arrow button (shown on the snapshot below) allows you to pick one of the recently used filters from a list.

Custom Filtering. At the far right end of the Filter Box you will find the Customize button that opens the Filter Builder dialog.



The dialog may take a few minutes to wrap your head around, but it is basically quite intuitive. The cursor turns into a "hand" when you pass it over items you can change. Click the mouse to see a pop-up list of available choices.

We are not going into further detail here on the assumption that if you can successfully build a custom filter expression using multiple sub-expressions and Boolean logic, you can undoubtedly figure out this dialog. You can save and reload filter expressions using the Open and Save As buttons.

Dynamic Summaries are described in their own chapter on Grid Summaries.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

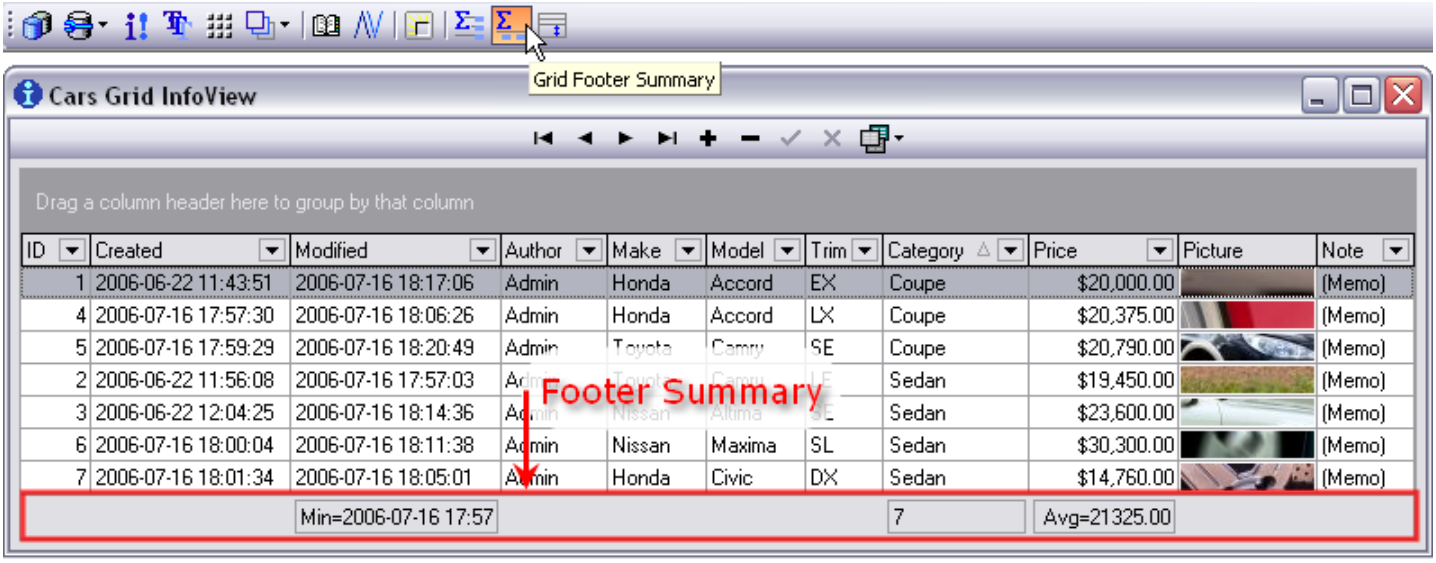
Grid Cell Auto Height. Normally, Grids display each record on one line, so any text that doesn't fit in a cell is simply cut-off. Clicking the Grid Cell Auto Height toggle button can turn On or Off the mode where each record occupies as much space as it needs to fully display its cell's contents. Such "oversized" contents may be strings of long text or pictures.

2.5 Grid Summaries

All Grids in Alventis support **Dynamic Grid Summaries**. Such Summaries can calculate some useful total/subtotal values for the columns they are attached to, such as: Count of records, Sum, Average, Min, Max of the column's values.

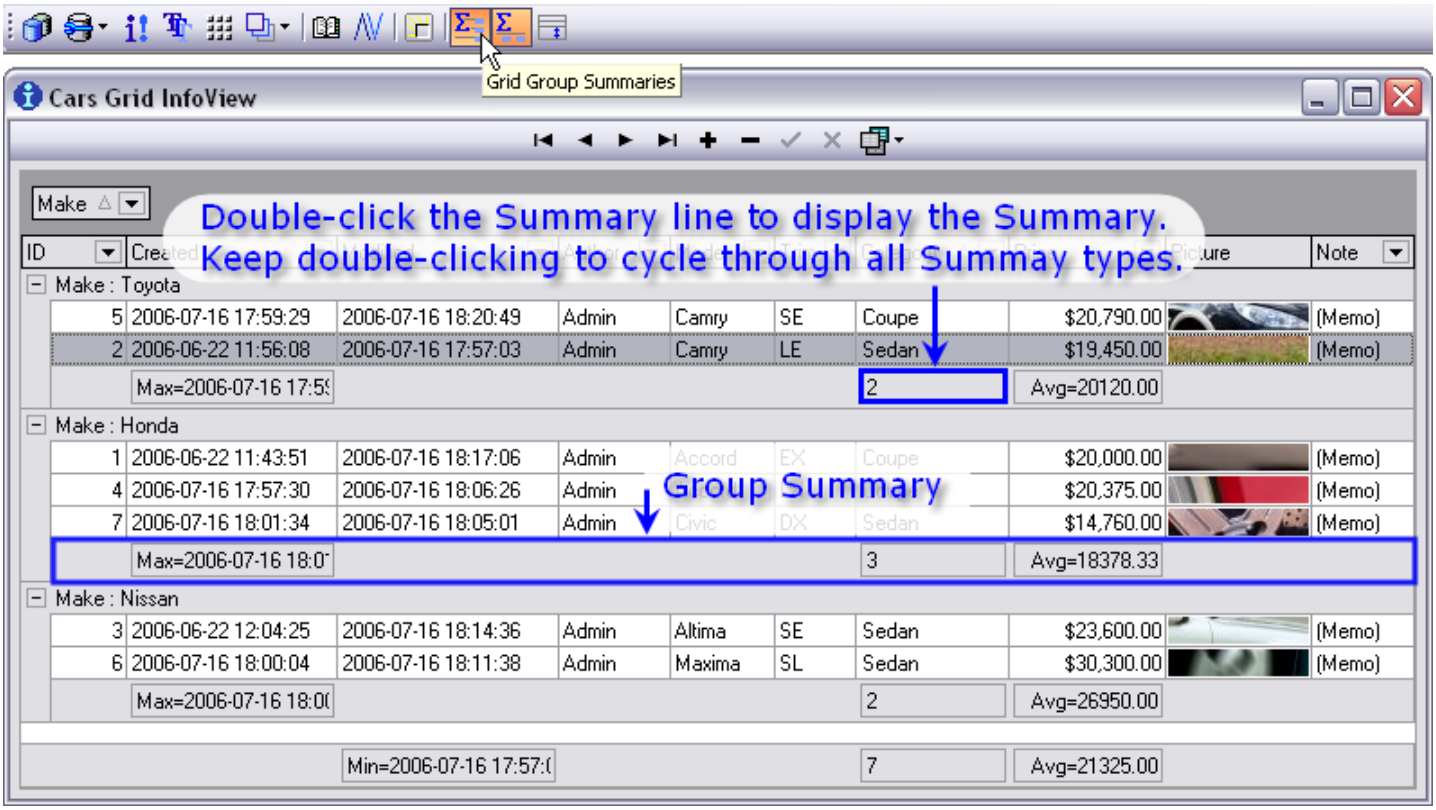
There are two kinds of Summaries:

Grid Footer Summaries are located at the very bottom of the grid and are calculated for the entire grid.


Grid Footer Summary
Cars Grid InfoView
Drag a column header here to group by that column
ID
Created
Modified
Author
Make
Model
Trim
Category
Price
Picture
Note
1 2006-06-22 11:43:51 2006-07-16 18:17:06 Admin Honda Accord EX Coupe $20,000.00 (Memo)
4 2006-07-16 17:57:30 2006-07-16 18:06:26 Admin Honda Accord LX Coupe $20,375.00 (Memo)
5 2006-07-16 17:59:29 2006-07-16 18:20:49 Admin Toyota Camry SE Coupe $20,790.00 (Memo)
2 2006-06-22 11:56:08 2006-07-16 17:57:03 Sedan $19,450.00 (Memo)
3 2006-06-22 12:04:25 2006-07-16 18:14:36 Sedan $23,600.00 (Memo)
6 2006-07-16 18:00:04 2006-07-16 18:11:38 Admin Nissan Maxima SL Sedan $30,300.00 (Memo)
7 2006-07-16 18:01:34 2006-07-16 18:05:01 Admin Honda Civic DX Sedan $14,760.00 (Memo)
Footer Summary
Min=2006-07-16 17:57
7
Avg=21325.00

Grid Group Summaries are located at the bottom of their respective grouping level and are calculated only for the records within that grouping level.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



In Alventis, you can create/modify/delete all such Grid Summaries whenever you wish, which is why we call them Dynamic.
You may have already seen how this Summary manipulation works in Reports. You should be aware of the fact that while providing mostly the same end result, the way it works is different between Reports and the rest of Alventis. We will be only discussing here how it works everywhere in Alventis – except the Reports, which have their own chapter dedicated to them.

Let's start with **Grid Footer Summaries**. Say, you have a grid, for instance, a Search Results grid, and you would like to create some Summaries for it. For example, have it tell you how many records it is displaying. The first step is to enable the Grid Footer Summary for it by clicking the button that carries the same name. This is a toggle button that simply shows or hides the Summary line at the bottom of the grid. Once we got the Summary line displayed, the rest is easy. Pick the column for which you would like to have some Summary and double-click the Summary line roughly within the vertical boundaries of that column.
This creates the first of possible Summary types using the first mathematical operation applicable to the data type of the column. Since all data types support at the very least counting their records, the first Summary type will always be Count, which is displayed as simply a number. If Count is not what you want, just double-click the Summary again, and the function will change to the next available, if any. So, double-clicking simply cycles through all possible Summary types in the following fashion:
None – Count – Sum – Average – Min – Max
If you keep double-clicking after the last summary type, it wraps around and starts again at the beginning.
Shift-double-clicking cycles in the opposite direction. Functions that are inappropriate for the selected column's data type are simply skipped, so for a string column, you'd only have None – Count to cycle through.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The **Grid Group Summaries** button toggles the Group Summary lines On/Off, and everything else works exactly the same way: just double-click to the Summary type you want.

Hiding Summary lines that contain some Summaries does not remove the latter, so you can temporarily hide them and then get your entire Summary line back intact with all its totals and averages.

For most grids, Alventis will preserve the Summary setup even if you close and re-open the form. The Dynamic Summaries are not shared between multiple users, if any: the Summaries you create are only visible to you, so you can set them up whichever way you want without worrying about your colleagues not liking your setup.

2.6 Memos

2.6.1 Memo Introduction

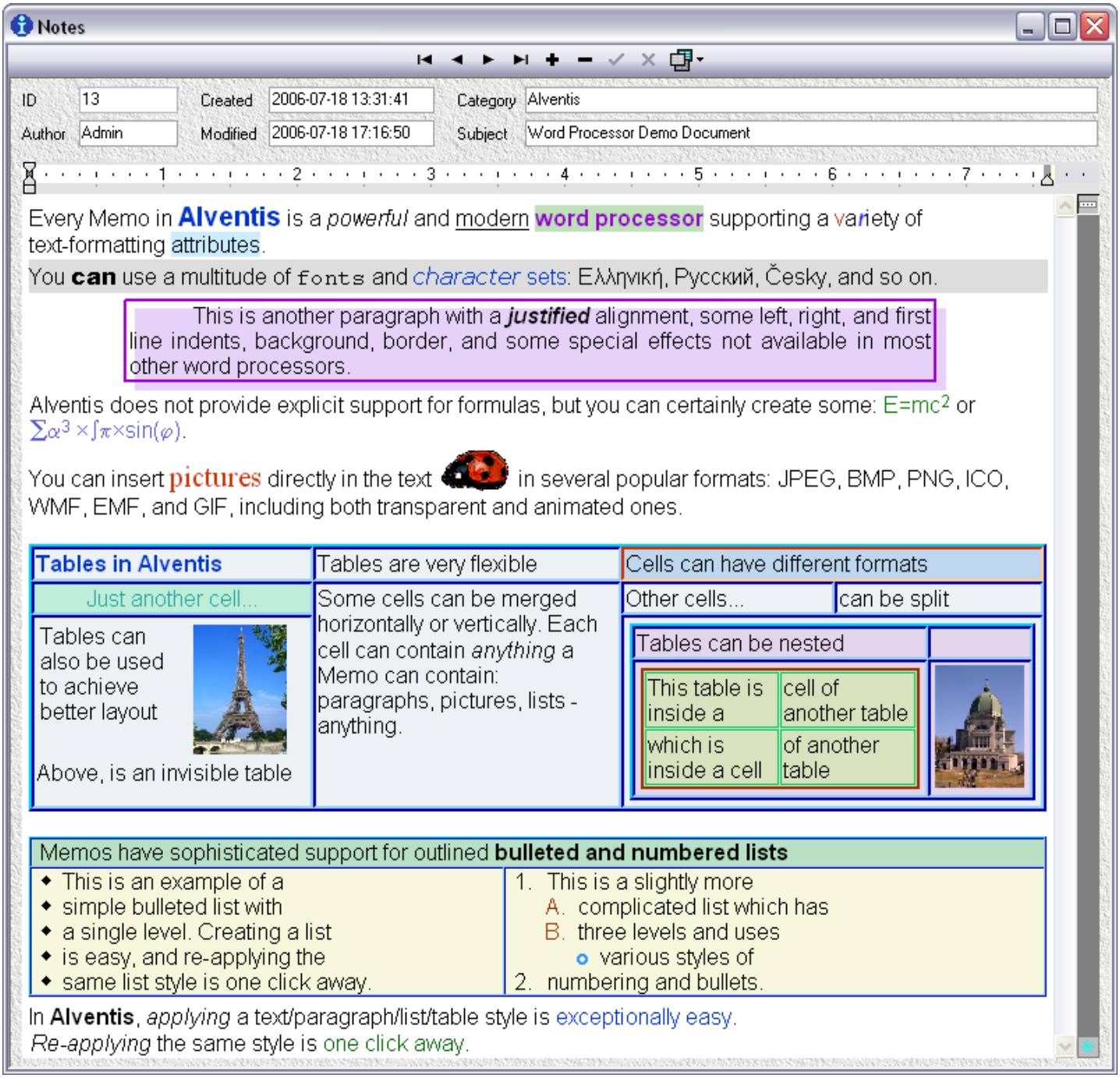
Aside from the usual assortment of Text Boxes and Combo Boxes and their like, Alventis supports Memo Boxes for editing free-form text. Each Memo is in essence a little modern word processor with support for complex formatting, text styles, embedded pictures, hierarchical tables, and so on. While we're at it, let's clear something up: there are two kinds of tables in Alventis that don't have anything to do with one another: data tables and memo tables. Data tables contain records and fields that hold your information. Memo tables are rectangular shapes with cells that are drawn in a document in a word processor – in our case, in a Memo. We will try to mention explicitly which kind of tables we are referring to whenever there's a chance of confusion. For the most part though, it should be fairly obvious which type we're talking about from the context. And since the entire Memos section of this Guide is dedicated to Memos, we will only be talking about memo tables here, so whenever we say "table" in this section, you should be thinking about "memo tables".

A Memo represents a single Memo-type field of a data table. Any given data table may have any number of such fields, two or more if necessary. Which means that you may encounter InfoViews with more than one Memo.

2.6.2 Things Unfamiliar

Anybody familiar with a word processor should quickly find him- or herself right at home with Alventis Memos. Which is why we won't go into excruciating detail on how to select text with a mouse or make the cursor jump to the beginning of the next word. Instead, we will start by listing things that may differ from the most well-known word processors or may not be obvious for other reasons.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Notes
ID 13
Created 2006-07-18 13:31:41
Category Alventis
Author Admin
Modified 2006-07-18 17:16:50
Subject Word Processor Demo Document
Every Memo in Alventis is a powerful and modern word processor supporting a variety of text-formatting attributes.
You can use a multitude of fonts and character sets: Ελληνική, Русский, Česky, and so on.
This is another paragraph with a justified alignment, some left, right, and first line indents, background, border, and some special effects not available in most other word processors.
Alventis does not provide explicit support for formulas, but you can certainly create some: E=mc2 or ∑α3 ×∫π×sin(φ).
You can insert pictures directly in the text in several popular formats: JPEG, BMP, PNG, ICO, WMF, EMF, and GIF, including both transparent and animated ones.
Tables in Alventis
Tables are very flexible
Cells can have different formats
Just another cell...
Other cells...
can be split
Tables can also be used to achieve better layout
Above, is an invisible table
Some cells can be merged horizontally or vertically. Each cell can contain anything a Memo can contain: paragraphs, pictures, lists - anything.
Tables can be nested
This table is inside a
cell of another table
which is inside a cell
of another table
Memos have sophisticated support for outlined bulleted and numbered lists
This is an example of a
simple bulleted list with
a single level. Creating a list
is easy, and re-applying the
same list style is one click away.
1. This is a slightly more
A. complicated list which has
B. three levels and uses
various styles of
2. numbering and bullets.
In Alventis, applying a text/paragraph/list/table style is exceptionally easy.
Re-applying the same style is one click away.

- No right margin. That's right, there sort of isn't one. Resizing the Memo extends it to the right (or shrinks it) – indefinitely. This obviously affects the way the text flows. The reason why this is the way things work has to do with printing. We won't go into detail here, but imagine a simple example. You have a table with 15 string fields (think ordinary Text Boxes) and a Memo. How shall we print out such records? We could of course put the string fields at the top followed by the Memo spanning the entire width of the page. Fine. But what if we wanted to lay things out differently. What if we wanted to place the 15 string fields on the left and the Memo on the right? Not a bad idea. Except that it means that it is the layout of the specific Report that will determine the width our Memo will be printed with. And we can have more than one Report with wildly different layouts. This is why we

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

cannot reliably predict what the "correct" width of our Memo should be. What if our table has two Memos? We may want to print them side-by-side – at least in some Reports. Well, you get the picture. Alventis focuses on the actual data rather than page layout, and as a consequence things that have to do with margins are very different from word processors you may be familiar with.

As a matter of fact, the are no margins in Memos at all. At least not until the Report stage. It is only then that you can lay things out on a page and decide how narrow or wide you want them. If you do want to know in advance how things will look when the Memo takes, say, the width of a whole page, is quite easy: just resize either the entire InfoView form or just the Memo portion of it (if it allows you to do so) in such a way that the Memo width becomes what you expect it to be one the final output page. That's one of the reasons the Ruler is at the top of every Memo. If it isn't, just click the little button at the top/right corner of the Memo to toggle it On.

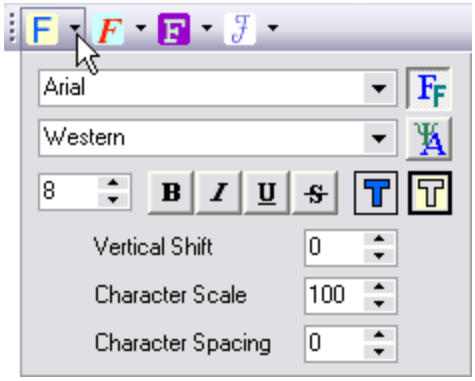
- No Real Left Margin Either. You can "pretend" there is one using the Ruler above the memo. Dragging the left margin at the point where the mouse cursor becomes a left-right arrow does indeed work. In the sense that you do end up with blank space on the left side of the Memo that looks like a margin. And that's precisely it: it only looks like one. This is not a real margin that will be saved with the Memo or get printed. It's nothing but blank space. It does have a use though. When the mouse is in that blank space, clicking it selects a line of text and dragging it selects multiple lines – just as in most word processors. This is why you may want to still have a small "margin" there.

- InstaButtons. They are: Font, QuickStyle, all Color buttons, Paragraph Format, Bullets and Numbering, Symbol Picker, all Memo Table related ones, Picture Format, and Border Style. The most important and unique characteristic of InstaButtons is that you can have as many of each kind as you like. With ordinary toolbar buttons and menu commands, even if you create multiple instances of the same button, they all behave as one. For example, place two Italic buttons on the toolbar. Whenever one is depressed, so is the other. No matter how many you place on the toolbars, all of them are "linked" and represent the exact same thing, which is why nobody uses more than one in the first place. Most buttons in Alventis are such ordinary buttons that do not offer any benefit from appearing more than once. Except InstaButtons.

Each InstaButton has its "personal" attribute or set of attributes that is independent from all others, even other InstaButtons of the very same kind.

Take for example the Font InstaButton, and imagine we have two such Font InstaButtons side-by-side on the toolbar.

The first one may "contain" a full set of font attributes of, say, the Arial font, blue-on-yellow. The second one may represent Times New Roman Italic / red-on-turquoise.

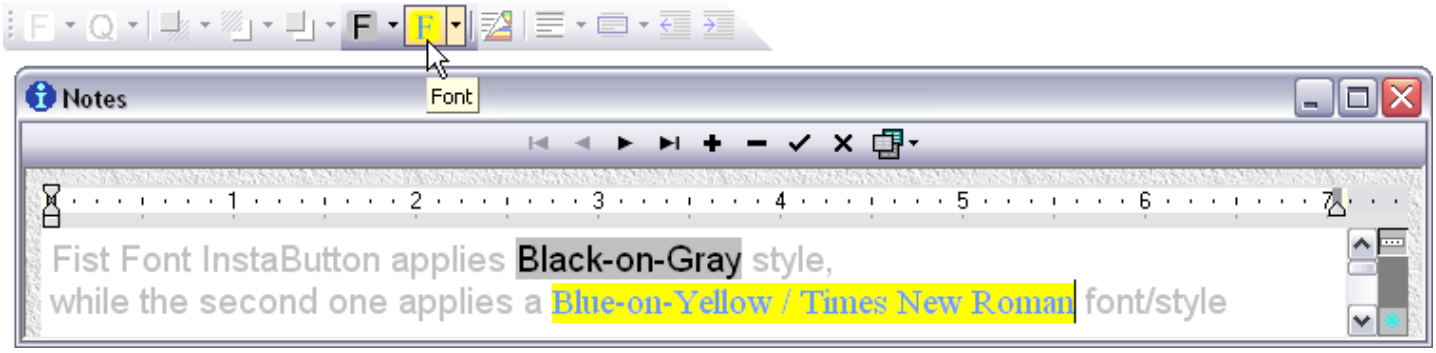


You will see in just a moment how you control which InstaButton has what set of attribute values, but the uniqueness of InstaButtons lies in the fact that the above two instances of the Font InstaButton behave as if they were completely separate bar items. The chapter on Toolbar Customization describes how you can customize the bars and menus and create as many copies or clones of an InstaButton as you like (right-click the toolbar if you are overly impatient).

Clicking an InstaButton immediately applies to the text whatever set of attributes it represents. Applying either of the two fonts for which there exist "dedicated" InstaButtons is therefore just one click

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

away.


Font
Notes
Fist Font InstaButton applies Black-on-Gray style,
while the second one applies a Blue-on-Yellow / Times New Roman font/style

Clicking on the little drop-down arrow next to an InstaButton opens a dialog box that allows you to edit that button's attributes. Most dropdown dialogs work in some ways like menus: they don't have an OK button, and they get closed automatically if you click anywhere outside them. More importantly though, most of them work in a "live" mode: any changes you perform in the dialog are immediately reflected in whatever selection you may have had in the Memo. For example: select some text and drop-down the Font dialog. It does not display the current font of the selected text, but rather the font of the InstaButton you dropped down. Change any attribute, e.g., turn Italics On: the selected text becomes italicized (while you're still in the Font dialog). Make the Bold button depressed: the selection becomes bold (in addition to whatever other attributes are selected in the Font dialog). And so on. Once satisfied, all you have to do is click wherever you want outside the dialog, perhaps somewhere in the text of the Memo: the Font dropdown closes, and the last set of attributes you "arrived to" has been assigned to the InstaButton – so that next time you need it, you could re-use the same set with a single click on that InstaButton. You can quickly assign a particular set of attributes to InstaButtons (e.g., Font, Bullets and Numbering) by placing the cursor somewhere in the formatting you like (or selecting it) and Shift-clicking the InstaButton.

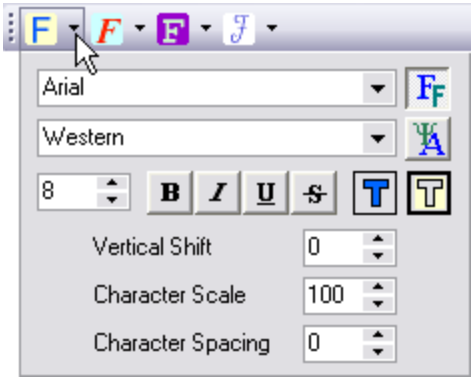
The QuickStyle InstaButton is special in quite a few respects. Basically, it allows you to apply or clear the set of text attributes of your choosing. It is described in detail in the QuickStyle chapter.

2.6.3 Font

The **Font** button is an InstaButton, so it follows the principles of operation of all InstaButtons. The Font dropdown dialog is depicted below. To open it, click on the little "down" arrow at the right edge of a Font button.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Let's examine its items. As you may have guessed, the combo boxes at the top are Font Name and Character Set. Character Sets (or Charsets) are described in a separate chapter dealing with International Issues. Below, we find a dense line of controls that are left-to-right: Font Size, Bold, Italic, Underline, Strikeout, Text Color, Background Color. These are followed by 3 boxes whose meaning would be enigmatic otherwise, so they are labeled: Vertical Shift, Character Scale, Character Spacing.
Vertical Shift is measured in % and it shifts text up or down (depending on the sign of the value) from the normal baseline. Alventis does not directly support superscript or subscript font styles, but you can emulate both using appropriate Vertical Shift values, perhaps also reducing the Font Size of the sub- or superscript.
Character Scale is also measured in % and it expands or contracts the entire selected text by the value specified.
Character Spacing is measured in pixels and is responsible for expanding or contracting only the spaces between characters – not the characters themselves.



That's it. Well, almost. There are also these two little buttons to the right of the Font Name and Character Set combo boxes. These are radio buttons: pressing one un-presses the other. Each of them, when pressed, indicates that its adjacent combo box takes priority over the other one. Here's how it works. Let's say you know which specific font you want to select by name. Be it Arial or Wingdings, you know the name you want. Make sure the Priority button next to the Font Name combo box is pressed. This fills the Font combo box with names of all fonts available on your machine. You can now select whichever font you want by name. Let's take another example though. Imagine you know the Charset you want to use, e.g., Greek. If you want to type something in Greek, Charset certainly takes priority: you want your text come out with proper characters, and what specific typeface (font name) they belong to is secondary. If you were using the first setup where Font Name had priority, you'd have to either know or guess the name of a font that supports the Greek Charset (many do, but not all). Failing that, you'd have to go through them one-by-one in search of the right one. Not so with our dialog. Just click on the Priority button next to the Charset combo box. Now it is this combo box that is filled with all Character Sets available on your computer. Pick the one you like – and the Font Name combo box will only display the names of fonts that were determined to support your chosen Charset.
You may not have too much need for this feature if you are not in a habit of typing multilingual text. But you may also find it useful if you are trying to go through, say, a list of fonts that support the Symbol Charset.

In closing, we'd like to note that whenever one of these combo boxes ends up containing a single item (be it Font or Charset), it becomes disabled as an indication to you that there's no choice available.

2.6.4 QuickStyle

QuickStyle is an InstaButton, so it follows the principles of operation of all InstaButtons. The few exceptions that are unique to QuickStyle buttons will be listed below.

The QuickStyle dropdown dialog is depicted below.
At the first glance, there's really not much to it: we have the familiar array of Font Attribute controls, accompanied by the somewhat less familiar Font/Charset Priority buttons. We also have something not

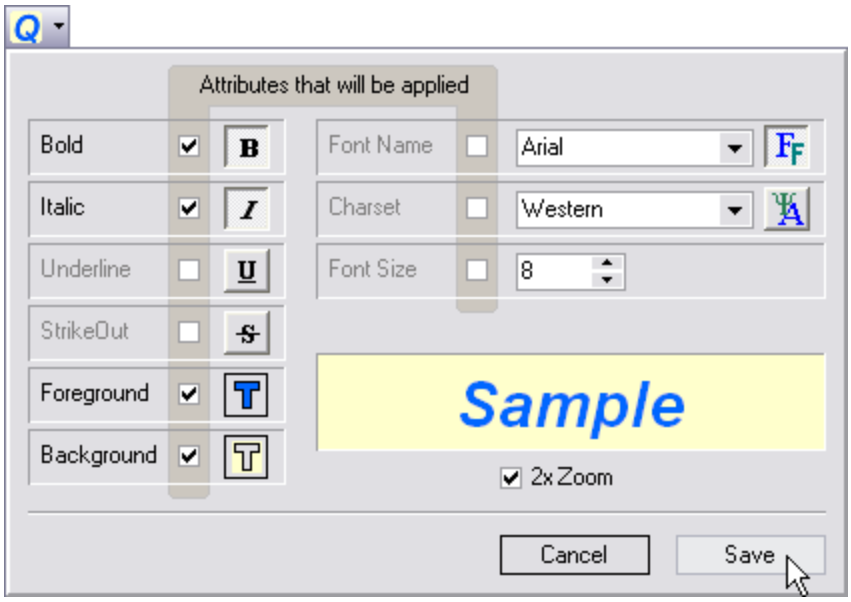
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

found in other InstaButtons: a preview that can be zoomed-in if you check the little checkbox under it, and amazingly, even Cancel and Save buttons.

The most important feature of this dialog though is the array of checkboxes next to every font attribute. These checkboxes specify which attributes will be applied to the text.
This should become clearer from an example.

Imagine we want to emphasize some portions of a long document in a particular way. Say, we want them to appear Bold, Italic, Blue Text on Yellow Background. Whatever other attributes the text may have, we want to leave them unchanged. Font name, Charset, size, among other things. Using ordinary formatting tools would be rather time-consuming: we'd need to select every portion of interest of the text and apply one-by-one the right attributes to it.

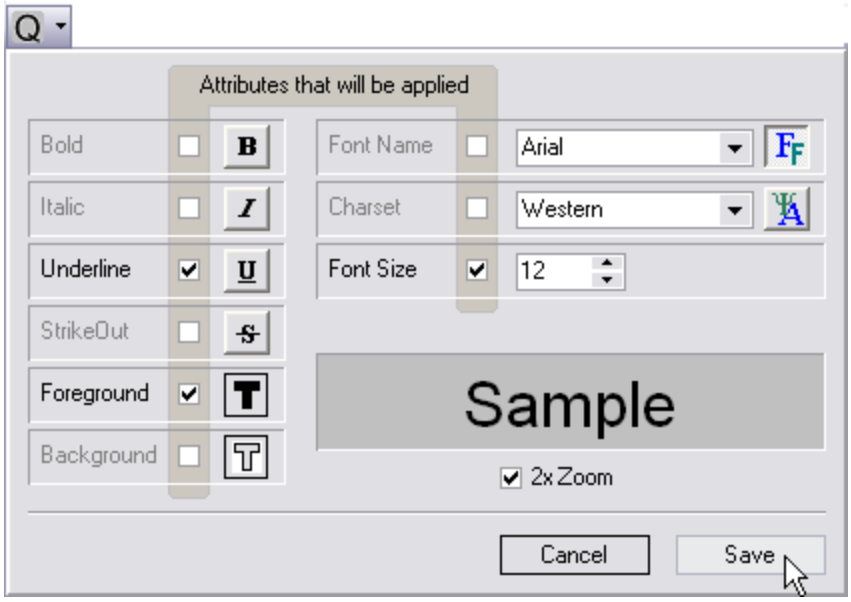
QuickStyle to the rescue! Let's set it up like so: Bold button depressed and checkbox next to it checked (each checkbox gets automatically checked as soon as you change its corresponding attribute). Same for Italic. Text Color set to Blue (and matching checkbox got checked, right?). Ditto for the Yellow Background Color. All other checkboxes must remain unchecked. What this tells this QuickStyle button is this: "Whatever text you are formatting, apply whatever we selected for the attributes that have a checkbox checked next to them (B, I, both Colors), but leave everything else alone".



Click Save to update the QuickStyle – and now you can use the new "tool" you have just created: simply select the text you want and click the QuickStyle button. One click. Four operations.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Another example. Let's say we have some text with a wild assortment of formatting: fonts of all kinds and sizes, some Bold, some Underlined, with some strange colors here and there. Imagine we want to clean things up a bit by removing all Underlining, setting all font sizes to 12 points, and setting text color to Black. Sure, we could achieve what we want one step at a time, but this may quickly get tedious. We'll set up our QuickStyle InstaButton as follows: Underling Off (button up) and corresponding checkbox checked, Font Size 12 and checked checkbox, Text Color set to Black and corresponding checkbox checked. All other checkboxes are left unchecked.



Click Save to update the InstaButton with our new setup. The only affected attributes will be those that had checkboxes checked next to them, and what these attributes will be set to is determined by the state of the corresponding control.

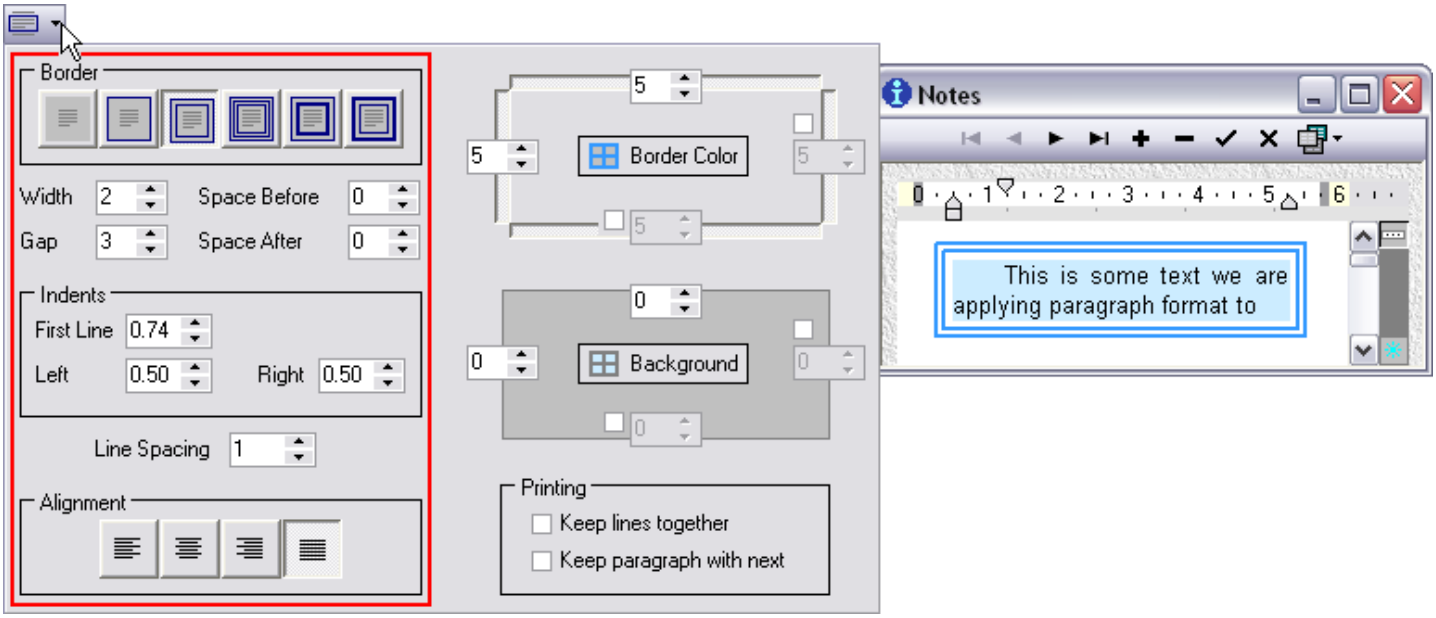
Each QuickStyle button attempts to give you as faithful a feedback as it can as to what its setup is. This is reflected in both the preview in the dialog and on the button itself. However, it cannot represent everything. For example how would it show that it is set up to not touch either the Text or the Background Colors? We simply chose two neutral shades of gray. Or how could it differentiate between you setting Underlining to off (U is up and checkbox checked) vs. you just not touching Underlining (U's state doesn't matter and checkbox unchecked)? There simply is no way to correctly represent all possible combinations. So, while a picture may usually be worth 1000 words, in this particular case, you may want to simply review a QuickStyle button's setup by dropping down its dialog.

2.6.5 Paragraph Format

The **Paragraph Format** button is an InstaButton, so it follows the principles of operation of all InstaButtons. It gives you direct access to all aspects of paragraph formatting in a single dialog.

The Paragraph Format dropdown dialog is depicted below.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Let's examine its items.
The Border radio button group in the top/left corner of the dialog allows you to select the type of paragraph border you would like. A few controls in this dialog only make sense if there is some border, so let's start by selecting a double-line border style to make sure we can play with everything this dialog has to offer.

Below the Borders radio group we find 4 SpinEdit boxes.

- Width specifies the width of the narrow lines that make up the border. The thicker lines used in the last two border styles are based on the thinner lines, so their widths are calculated automatically. Width is measured in pixels.
- Gap is the distance between lines of multi-line borders (in pixels). It obviously only applies to border styles that use more than a single line, so you will find this SpinEdit disabled for the single-line border style.
- Space Before specifies any additional amount of space (in pixels) that you may want to leave above the paragraph. This makes the paragraph distance itself from the preceding paragraph by the specified amount of space.
- Space After does the same thing, but adds space below the paragraph.

We now get to the "Indents" group of SpinEdit boxes. They specify what indents the paragraph is to have as depicted below.
While we are on this subject, it is worth noting that you can set indents using the Ruler by simply dragging around the corresponding indicator.
Finally, you can also increase or decrease indents to the next or previous Tab Stop using the Indent and Outdent toolbar buttons.

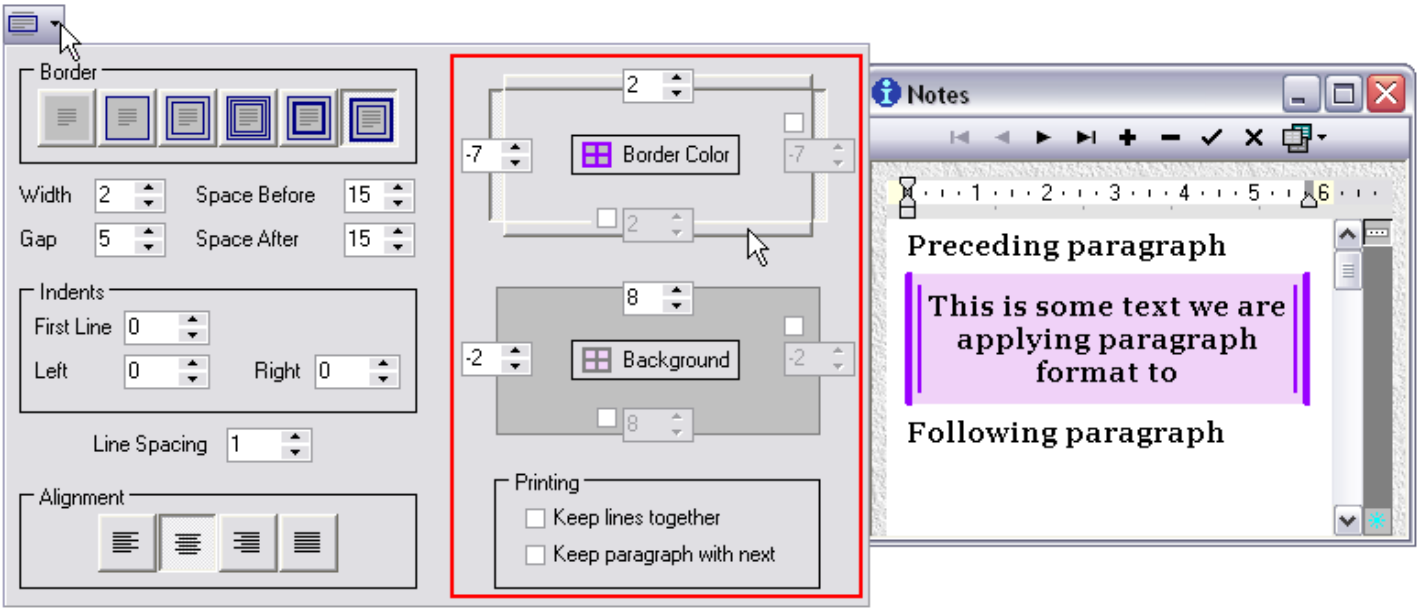
All indents are measured in the units of the active Memo, which could be either inches or centimeters. You can change the units from the Tabs dialog. You can also set your preferred default units globally from the Settings dialog.

The next SpinEdit specifies Line Spacing. A value of 1 means single line spacing, 2 means double spacing, and so on. You can specify fractional spacing values, e.g., 1.5 or 0.85.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The "Alignment" group of radio buttons specifies, well, the alignment of the text in the paragraph: Left / Center / Right / Justified. Time to move on to the fun stuff!

On the right-hand side of the dialog we find two similar groups of controls.



The first one has to do with the appearance of paragraph borders. The button in the middle is candid enough to tell us what it does: it specifies the border color. Easy.

What may look like a rectangle made of sticks is precisely that: it's a representation of the paragraph border made up of the 4 line segments represented by toggle buttons. Each button allows you to turn its respective border edge On or Off. Just try clicking any of them to see how it works.

The SpinEdits in the middle of each border edge are border Bias values. Sounds fancy but it's quite simple: each of them specifies by how many pixels their respective border edge should be moved inward or outward. For example, let say we wanted to make the left and right edges of the border a bit further from the paragraph text than they are with a bias value of 0. Just set the Left Bias value to some small positive number (e.g., 2 or 3) and see how the border moved away from the text on both left and right sides. Normally, you'd want borders to be symmetrical, which is why modifying the Left Bias automatically keeps the Right Bias in sync. But if you want to control bias values individually, just check the checkbox next to the Right Bias SpinEdit: you can now change Left and Right biases independently from one another. Same goes of course for the Top/Bottom biases. You can use negative bias values too: they move the border edges inwards, into the paragraph.

Moving on, we get to a similar-looking arrangement of controls, except that this group is responsible for the appearance of the paragraph Background. The button in the middle sets the color, but we won't tell you of what. The SpinEdits that surround it work in precisely the same way as their Border Bias counterparts described earlier. The Background Bias values move the background rectangle around.

Finally, in the bottom/right corner of the dialog we find a couple of checkboxes in a Printing group. Keep lines together, if checked, will attempt to prevent the paragraph from being cut in half by a page break. Such a paragraph will simply move to the next page, which should keep it in one piece.

Keep paragraph with next is similar. It will try to make sure this paragraph and the one that follows it

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

end up on the same page.

Needless to say, these two settings only apply to the printing stage of the game, so you will not see any immediate effect of applying these setting to a Memo paragraph.

2.6.6 Color Picker

All Color buttons are InstaButtons, so they follow the principles of operation of all InstaButtons.

The Font Color, the Background Color, and the Paragraph Background buttons invoke the Color dropdown dialog called Color Picker which is shown on the right.

As colorful as it may be, it is very simple. All you do here is pick a color from the available palettes. At the top, you have 8 pages, each with 2 palettes: the color palette at the top and the grayscale one beneath it. Further down, there's a Custom Colors palette. Finally, in the bottom-left corner there's a shy micro-palette with a single not-even-a-color called None. More on these two later.

At the bottom you will see a preview that gets updated as you move the mouse over the palettes. Wherever the mouse pointer changes to a crosshair, you can click on the underlying color to select it – and that's usually all there is to it! The moment you select a color, it is assigned to whatever button invoked the Color Picker, and the dialog is automatically closed.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

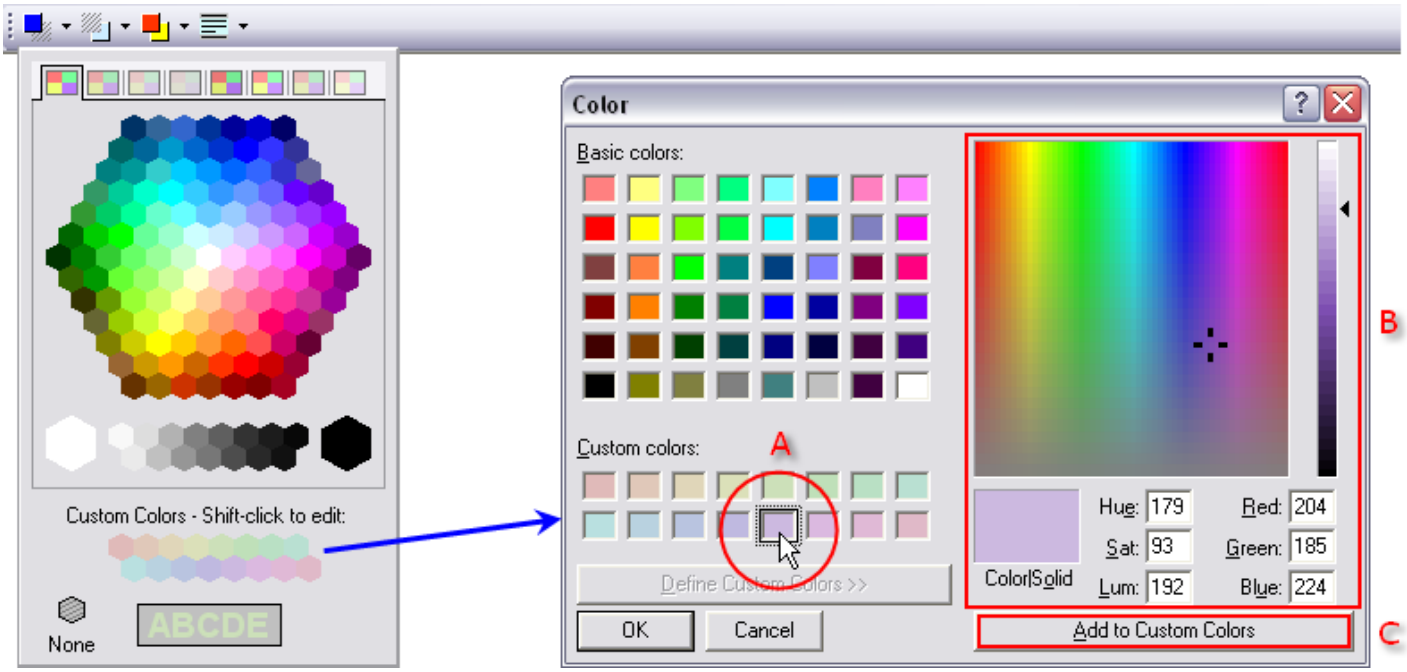
The only exception is the Color Picker invoked by the Both Colors button. This button is special: it allows you to set both colors at once (and apply them both to text in one shot). Such being the case, this Color Picker must give you an opportunity to select just the right pair of colors, so you can keep selecting and re-selecting both until you are satisfied. You select the foreground color by clicking the palette, and the background by right-clicking it. Once you are happy with your selections, you have to explicitly hit the OK button in the corner.

And what about that "None" color? Well, for starters, it's not even a color. It is only useful under some rare circumstances and only for styles of some items. One example is text background color in Memos. Let's say we set the background of a single word in a paragraph to Yellow. Imagine that you later decide to change the background of the entire paragraph that contains this word to Green. At this point in time you decide you no longer need or like the Yellow highlighting of that word, so... well, maybe you just set its background to Green too. Okay, things look just fine. However, what if you later decide to change the background of the paragraph to something other than Green, say, Blue. The paragraph becomes Blue, but that word in the middle of it has its own "personal" background, so it remains Green. Not so good. The solution is simple: set the background color of this word to None. This simply tells whatever object you are colorizing to use the default color, which is typically interpreted as transparent or "no color".

Some objects or items do not "understand" the None color, and interpret it "their way", which is typically as Black. If this happens, just don't apply the None color to such items (e.g., Memo text (foreground) color).

Custom Colors Palette. Merely using or applying its colors is no different from the other palettes: just click (or right-click for background) the color you want. What makes this palette special is that you can customize the 16 colors it contains. To do so, Shift-click anywhere within the palette. This opens the standard Windows Color dialog.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The 16 Custom Colors appear in the lower-left area of the dialog. You can change one or more of them using the following procedure:
A) Click the Custom Color you want to change
B) Use the right half of the dialog to select the new replacement color. Note that you can also click one of the Basic Colors listed in the upper-left quarter of the dialog to give yourself a "starting point": clicking a Basic Color simply sets all controls in the right half of the dialog to matching values.
C) Click the Add to Custom Colors button to assign the selected color to the color cell you have selected in step "A" above.

Alternatively, if you want to set one Custom Color after another in succession, you can start with the above A,B,C procedure, and then continue with just B,C as many times as you want. Doing so will keep adding the colors of your choosing to the next color cell – one after another.
Once satisfied with all the 16 Custom Colors, you can save them all to the Alventis Color Picker by hitting the OK button. You can of course discard your changes by hitting Cancel.

2.6.7 Bullets and Numbering

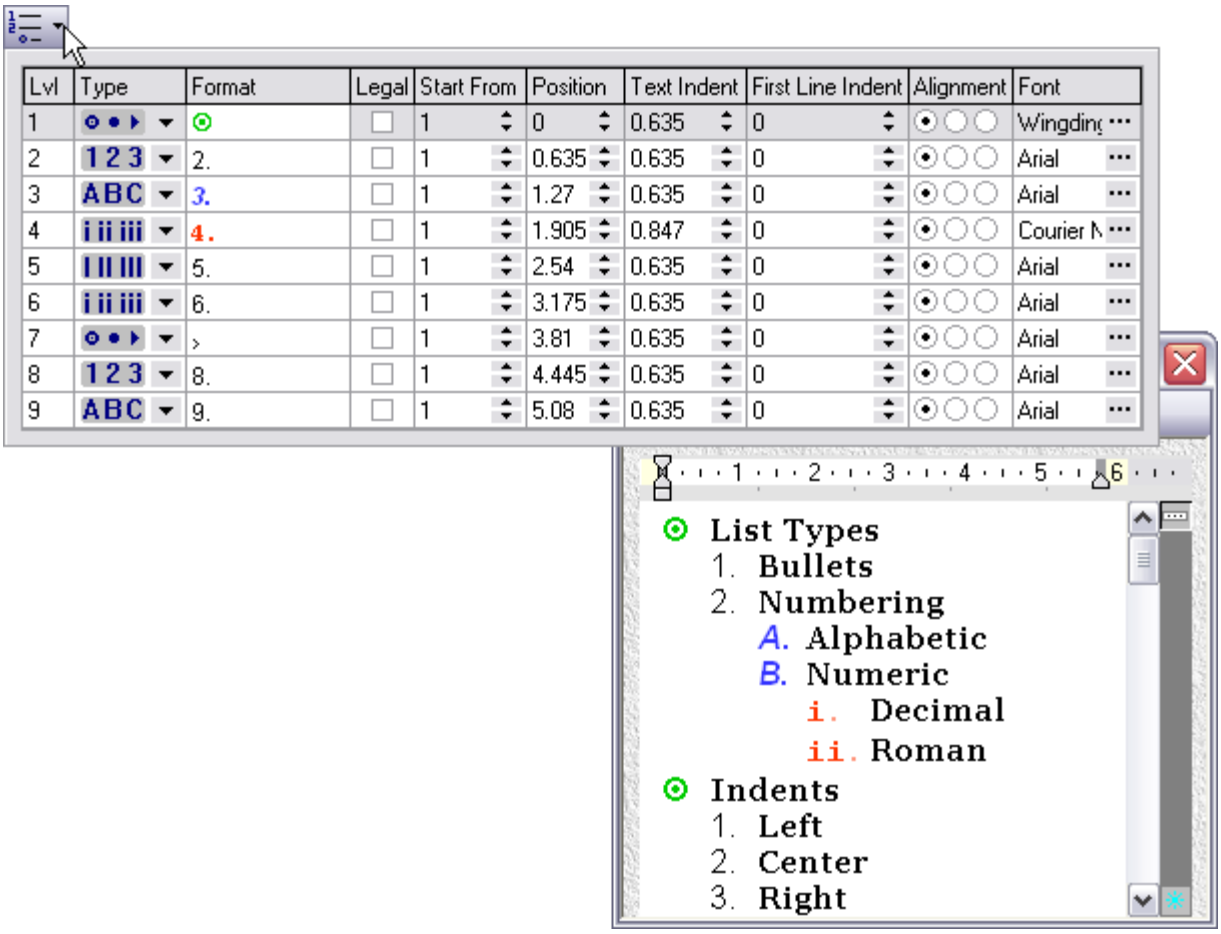
The **Bullets and Numbering** (B&N) button is an InstaButton, so it follows the principles of operation of all InstaButtons.

Alventis supports the following B&N features:
- Outline B&N of 9 levels
- Six types of B&N formats
- All B&N formats are completely user-definable
- Legal-style numbering on a per-level basis
- Selection of starting number for each level
- Per-level position/indents control
- Left/Center/Right alignment for the bullet/number

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

- Full control over the font of the bullet/number

And we've just completed a quick left-to-right run through the entire B&N dropdown dialog! That's right, the entire power and flexibility of Alventis' B&N is right at your fingertips – all in a single dialog box.


Lvl Type Format Legal Start From Position Text Indent First Line Indent Alignment Font
1 2. 1 0 0.635 0 Wingding
2 1 2 3 2. 1 0.635 0.635 0 Arial
3 ABC 3. 1 1.27 0.635 0 Arial
4 i ii iii 4. 1 1.905 0.847 0 Courier N
5 I II III 5. 1 2.54 0.635 0 Arial
6 i ii iii 6. 1 3.175 0.635 0 Arial
7 › 1 3.81 0.635 0 Arial
8 1 2 3 8. 1 4.445 0.635 0 Arial
9 ABC 9. 1 5.08 0.635 0 Arial
List Types
1. Bullets
2. Numbering
A. Alphabetic
B. Numeric
i. Decimal
ii. Roman
Indents
1. Left
2. Center
3. Right

All 9 levels work the same way, so we will use the first level as an example in the following discussion, but everything we say applies to them all. Let's look at the columns of the B&N grid one at a time.

The Lvl column is simply the level number.

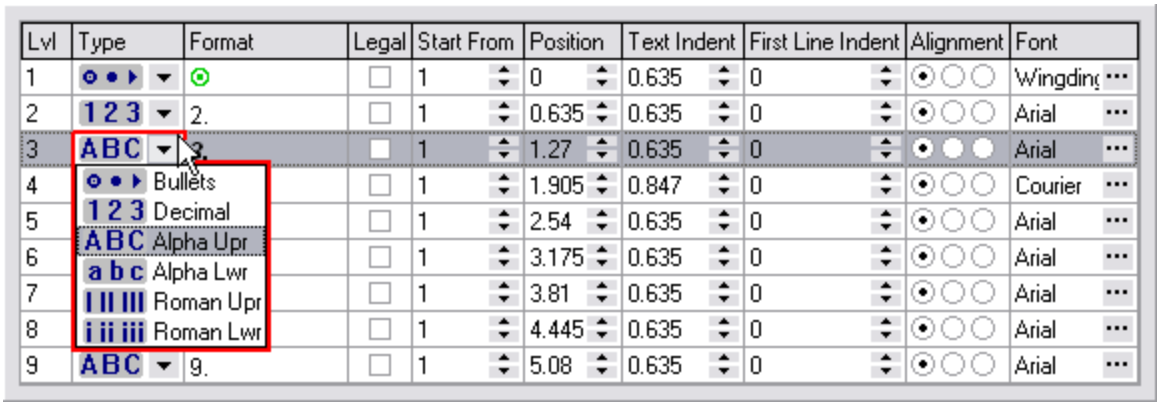
The Type column is much more interesting. It offers you a choice of 6 types of B&N:

- Bullets
- Decimal Numbers
- Alphabetical Uppercase
- Alphabetical Lowercase
- Roman Numerals Uppercase
- Roman Numerals Lowercase

These Types fall into two groups. The sole first Type finds itself in the Bullets group. The remaining 5 end up in the Numbering group. All 5 function in basically the same way, so we will cumulatively refer to them as "Numbering" – even though A/B/C are not usually considered numbers. We'll use both Decimal Numbers and Alphabetical Uppercase Types in our examples, but they apply equally well to the

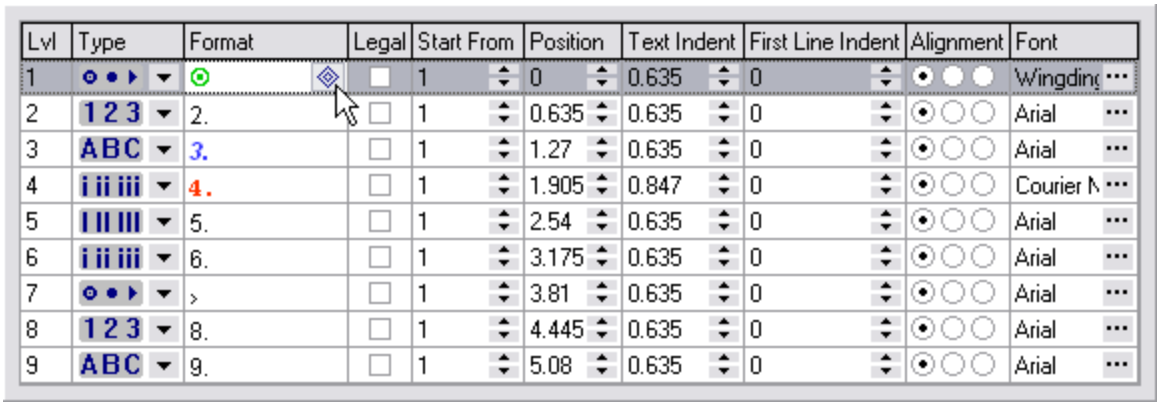
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

remaining members of this group.



Before we get deep into the intricacies of B&N formats, a quick overview of B&N is in order. You are probably quite familiar with bulleted lists from other word processors. Not much is different here. Except perhaps a few little things.
A Bullet or Number is not limited to a single character. You can enter as many as you like – and they will all be treated as a single bullet or number. You may not find yourself using multi-character bullets very often, but this feature is used just about all the time with numbers. The dot or parenthesis after the number is the most common example. You may want to have your numbers fully-enclosed in parenthesis like so: (1), (2), (3). You can add any text you like on either side, e.g., Chapter 1, Chapter 2, etc.

A Bullet or Number can use its own font. This includes all font attributes, except for the Background Color: Font Name, Charset, Text Color, you name it.

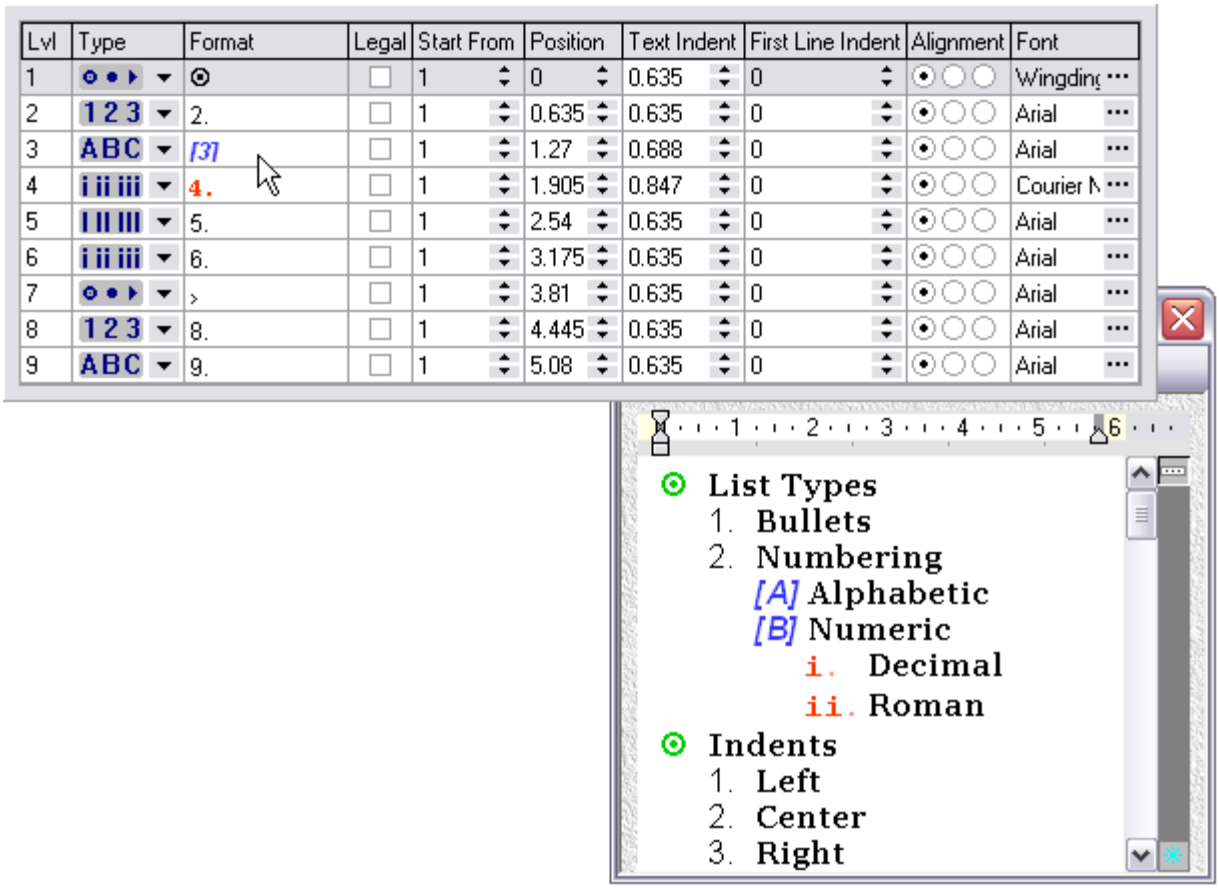


To facilitate entering special symbols you may want to use for your Bullets (or perhaps as part of your Number Format strings), you have access to the Symbol Picker right from the Format edit box. Just get into a Format box, place the cursor where you want to insert the symbol(s), and click the button with a diamond-shaped symbol. The Symbol Picker's operation is fully described in its own chapter.
This said, there remains nothing special to tell you about Bullets. So, let's move on to Numbers.

Formatting Numbering levels is very easy. All you need to do is type whatever you would like the level to look like. To have a list numbered with 1. 2. 3. format, all you need to type is "1." in the first level. To have the 1) 2) 3) format, you'd type "1)". To have the fancy Chapter 1. Chapter 2. Chapter 3. format – you guessed it – you simply type "Chapter 1.". There's only one trick to this magic. That "1"

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

you kept typing in every Format is special. Whenever Alventis sees a number in a level Format that is less than or equal to the Lvl number, it figures that this is where you want the actual Numbering digit or letter to appear. So, for instance, if we were working with the third level, and wanted it to look like [A] [B] [C], we'd set level 3 to have the Alphabetical Uppercase Type and we'd set its Format to "[3]". Note that we are now using the number "3" since it is the third level we're working with. Note also that we always use the level number ("3" in this case) regardless of which Numbering Type is in effect (e.g., Alphabetical in the above example).


Lvl Type Format Legal Start From Position Text Indent First Line Indent Alignment Font
1 0 0.635 0 Wingding
2 1 2 3 2. 1 0.635 0.635 0 Arial
3 ABC [3] 1 1.27 0.688 0 Arial
4 i ii iii 4. 1 1.905 0.847 0 Courier N
5 I II III 5. 1 2.54 0.635 0 Arial
6 i ii iii 6. 1 3.175 0.635 0 Arial
7 › 1 3.81 0.635 0 Arial
8 1 2 3 8. 1 4.445 0.635 0 Arial
9 ABC 9. 1 5.08 0.635 0 Arial
List Types
1. Bullets
2. Numbering
[A] Alphabetic
[B] Numeric
i. Decimal
ii. Roman
Indents
1. Left
2. Center
3. Right

And what if you want to include a literal character "3" in the third level Format string? Just precede such characters with a backslash "\". For example, a format "The \3rd Man, Scene 3." would produce a list appropriate for a screenplay of that film whose 3rd level would look something like this:
The 3rd Man, Scene 1.
The 3rd Man, Scene 2.
The 3rd Man, Scene 3.

You may have noticed that we stated earlier that the number used in the Format string may be less than or equal to the Lvl number. This allows you to produce multi-level numbering strings by repeating the previous level's number in the Format string of the current level. Alright, we didn't understand this explanation either, so an example is in order. Let's play with a 2-level list where the first level has the Alphabetical Uppercase type, the second level has the Decimal Number type (just to better distinguish them), and whose Format strings are respectively "1." and "2.". We will get a list that looks something like this:

A. First level and first item

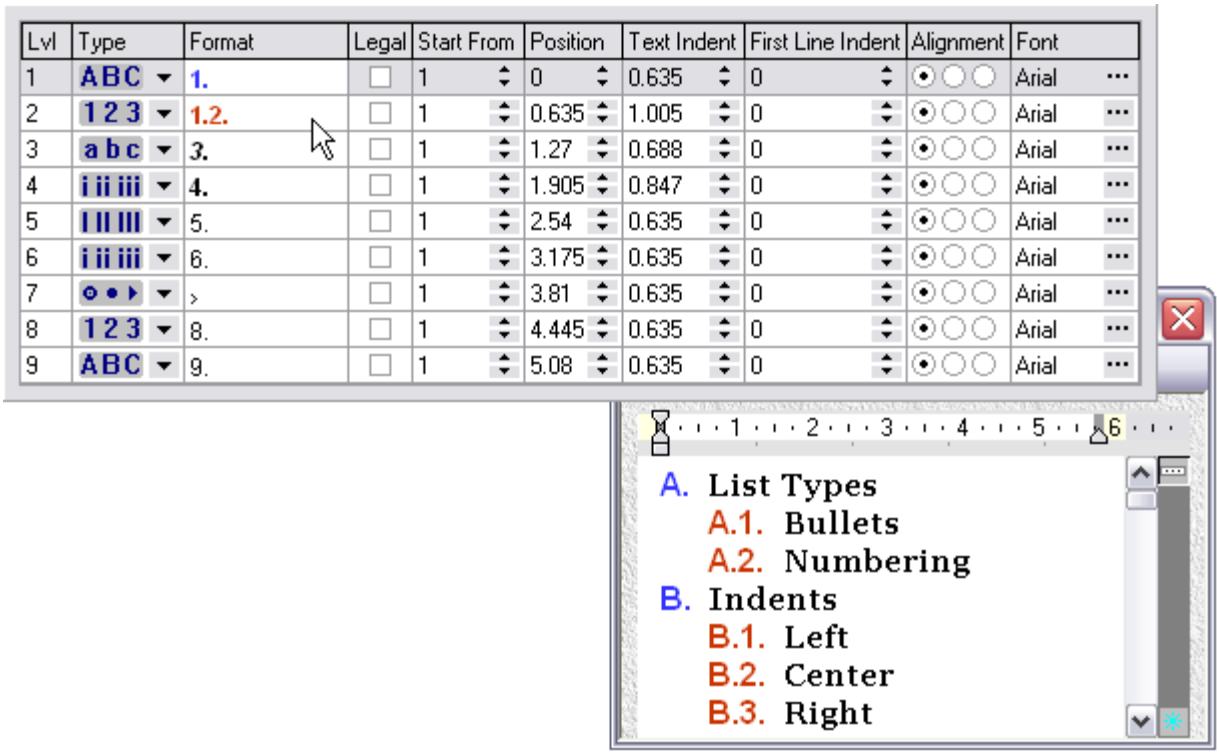
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

B. First level and second item
   1. Second level and first item
   2. Second level and – you guessed it – 2nd item
C. Back to First level, third item

No surprises so far. Let's now set the second level's Format string to "1.2.". Our list changes to:

A. First level and first item
B. First level and second item
   B.1. Second level and first item
   B.2. Second level and – you guessed it – 2nd item
C. Back to First level, third item

As you can see, the second level has substituted the "1" and "2" in its Format string by the appropriate list values.

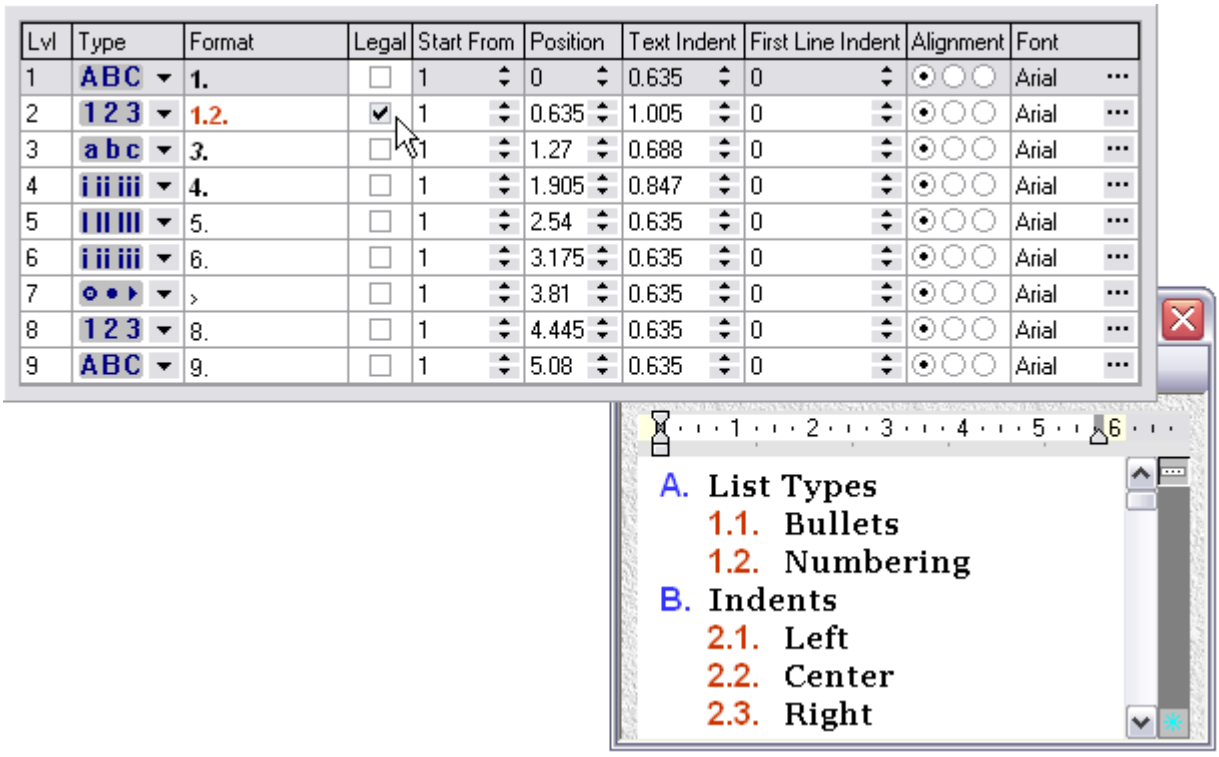

Lvl Type Format Legal Start From Position Text Indent First Line Indent Alignment Font
1 ABC 1. 1 0 0.635 0 Arial
2 1 2 3 1.2. 1 0.635 1.005 0 Arial
3 a b c 3. 1 1.27 0.688 0 Arial
4 i ii iii 4. 1 1.905 0.847 0 Arial
5 I II III 5. 1 2.54 0.635 0 Arial
6 i ii iii 6. 1 3.175 0.635 0 Arial
7 › 1 3.81 0.635 0 Arial
8 1 2 3 8. 1 4.445 0.635 0 Arial
9 ABC 9. 1 5.08 0.635 0 Arial
A. List Types
A.1. Bullets
A.2. Numbering
B. Indents
B.1. Left
B.2. Center
B.3. Right

While we're at it, let's discuss some Legal issues. Let's enable (check) the Legal checkbox for the second level. Our list becomes:

A. First level and first item
B. First level and second item
   2.1. Second level and first item
   2.2. Second level and – you guessed it – 2nd item
C. Back to First level, third item

The "2.1." and "2.2." numbering you see is Legal numbering. Basically, Legal-style numbering replaces any Alphabetical or Roman Number previous-level numbering by equivalent Decimal Number (so, in the example above, "B" became "2").

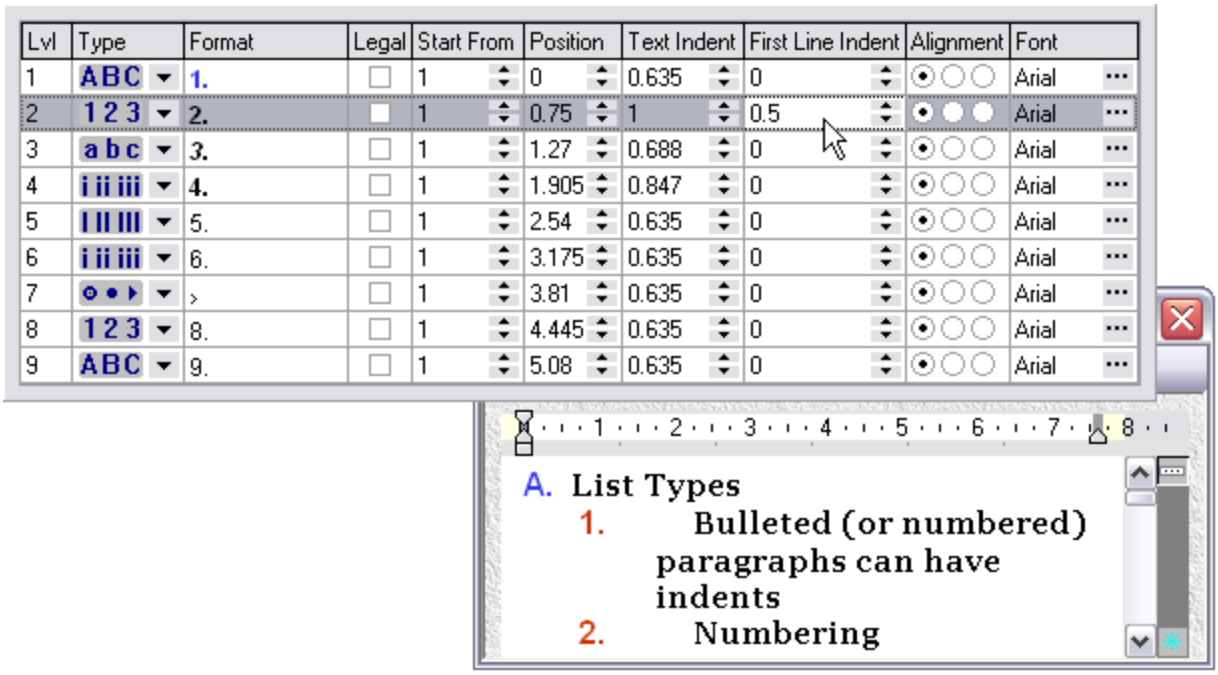
Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Lvl Type Format Legal Start From Position Text Indent First Line Indent Alignment Font
1 ABC 1. 1 0 0.635 0 Arial
2 1 2 3 1.2. 1 0.635 1.005 0 Arial
3 a b c 3. 1 1.27 0.688 0 Arial
4 i ii iii 4. 1 1.905 0.847 0 Arial
5 I II III 5. 1 2.54 0.635 0 Arial
6 i ii iii 6. 1 3.175 0.635 0 Arial
7 › 1 3.81 0.635 0 Arial
8 1 2 3 8. 1 4.445 0.635 0 Arial
9 ABC 9. 1 5.08 0.635 0 Arial
A. List Types
1.1. Bullets
1.2. Numbering
B. Indents
2.1. Left
2.2. Center
2.3. Right

The Start From column is easy. It specifies what number or letter the level should start counting from. If you wanted your list to number things 10. 11. 12. and so on, you'd simply set Start From to 10 for the corresponding level. Alphabetical numbering can also start from a letter other than A. You just need to specify the starting point numerically. So, if you want to start from C, you'd set Start From to 3. Same goes for Roman numerals.

The next three columns are Position, Text Indent, and First Line Indent. They are all measured in active units of the current Memo – either inches or centimeters.
Position specifies the position of the bullet/number relative to the left margin.
Text Indent specifies the distance from the bullet/number Position (as specified above) to where the text begins.
First Line Indent specifies precisely what the name suggests.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Lvl Type Format Legal Start From Position Text Indent First Line Indent Alignment Font
1 ABC 1. 1 0 0.635 0 Arial
2 1 2 3 2. 1 0.75 1 0.5 Arial
3 a b c 3. 1 1.27 0.688 0 Arial
4 i ii iii 4. 1 1.905 0.847 0 Arial
5 I II III 5. 1 2.54 0.635 0 Arial
6 i ii iii 6. 1 3.175 0.635 0 Arial
7 › 1 3.81 0.635 0 Arial
8 1 2 3 8. 1 4.445 0.635 0 Arial
9 ABC 9. 1 5.08 0.635 0 Arial
A. List Types
1. Bulleted (or numbered) paragraphs can have indents
2. Numbering

One thing you should be aware of is that the Ruler that sits above the Memo cannot be used to adjust B&N list formatting. The only way to control the layout of B&N lists is from the B&N dialog.

The Alignment column specifies how the B&N string is aligned relative to the specified Position: Left/Center/Right.

The Font column gives you full control over the font used for the B&N string. Just click the little button in that column – and you get the (hopefully somewhat familiar by now) Font dropdown dialog. The only attribute you can't set for lists is Background Color, which is why that control is disabled. Note that the entire Format string will use the same Font, but this should not limit your creativity too much.

You can clear B&N from the selected paragraphs by clicking the Clear Bullets/Numbering button.

A final word of warning. If you want to modify an existing list, you have to select the entire list before you start playing with the B&N dialog since it will apply its format to whatever your current selection happens to be.

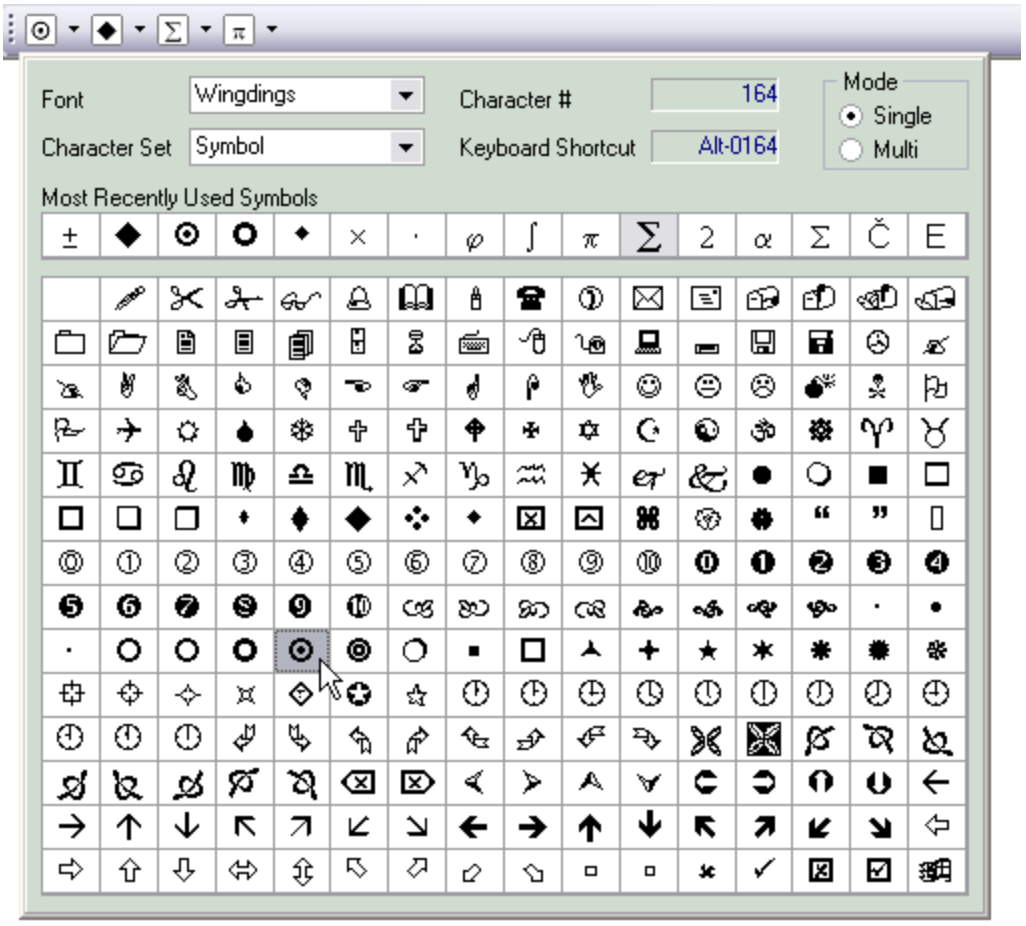
2.6.8 Symbol Picker

The **Symbol Picker** button is an InstaButton, so it follows the principles of operation of all InstaButtons.

The Symbol Picker dropdown dialog is depicted below.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Most of the dropdown is occupied by a matrix of symbols. You can select the desired Character Set and Font in the combo boxes in the top/left corner of the dialog. Since the main goal of the Symbol Picker is to select a symbol, the Character Set always takes precedence over the Font, i.e., the Font combo box always displays only the fonts that support the selected Character Set, so you may want to start by properly setting the latter.



Symbols you pick from the symbol matrix are added to the line of Most Recently Used Symbols above it. If the selected symbol is already there, it is not re-added.

The Character # box displays the ASCII number of the symbol as you pass the mouse over the matrix. The Keyboard Shortcut displays the shortcut you can use to enter the symbol under the mouse directly via the keyboard. The numbers you may see after the Alt- prefix must be entered on the numeric keypad.

Mode. The Single mode will enter the symbol you select into the destination text and dismiss the Symbol Picker. The Multi mode will send the symbol into the destination text but will leave the Symbol Picker open. This allows you to enter more than one symbol in sequence, thus turning the Symbol Picker into a "virtual keyboard". This can be quite useful to "type" a word in a foreign language, for example. You can dismiss the Symbol Picker once you are done as you would any other InstaButton dropdown dialog: just click outside it or hit Escape.

The last symbol you select is always assigned to the invoking InstaButton, so re-using the last symbol is as easy as simply clicking the InstaButton. You can have as many Symbol Picker InstaButtons as you like of course – each with its own "favorite" symbol.

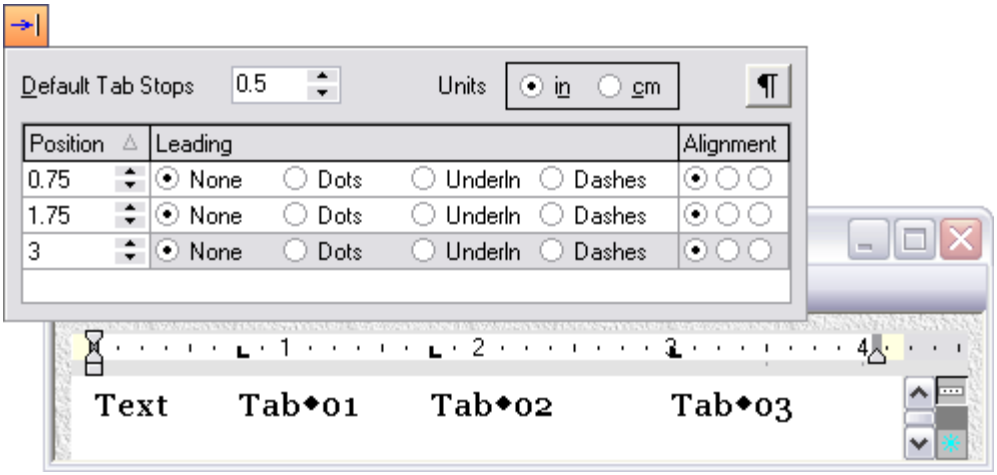
Finally, there's just one "twist" to the Alventis Symbol Picker: it is resizable. If you ever had trouble

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

making out the tiny symbols in other applications' symbol pickers, you will appreciate this feature.

2.6.9 Tab Stops

Memo paragraphs can have user-defined **Tab stops**. The way it works is rather similar to most common word processors. You can set Tab stops by clicking on the Ruler with the mouse. You can move Tab stops by dragging them along the Ruler. You can delete them by dragging them away from the Ruler. However, the Ruler only allows you to create the most common kind of Tab stop: the Left-aligned kind. If you need other kinds of Tabs or you want to specify their positioning numerically, you need the Tabs dropdown dialog.



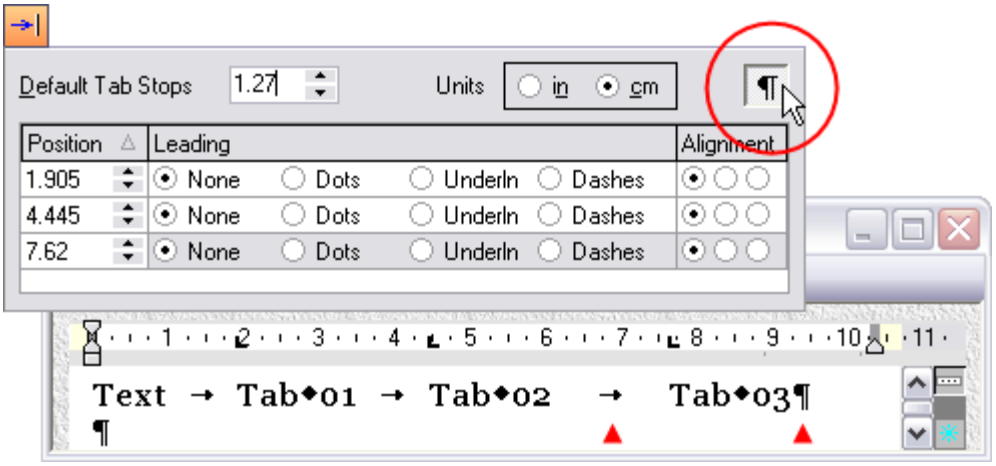
First, a little note. While it looks like an InstaButton dropdown dialog and in many ways functions like the rest of them, the Tabs button is not an InstaButton. It does not apply anything when you click on it (just opens the dialog), does not remember its "personal" settings, and it therefore doesn't make any sense to have more than one.

Every paragraph in a Memo starts out with Default Tab Stops. This is a single value that applies to the entire Memo – not to any particular paragraph. It will usually be equal to 1/2 inch or 1.27 centimeters, depending on your unit of choice. Unless you change it to something else, and the Tabs dialog is where you do so.

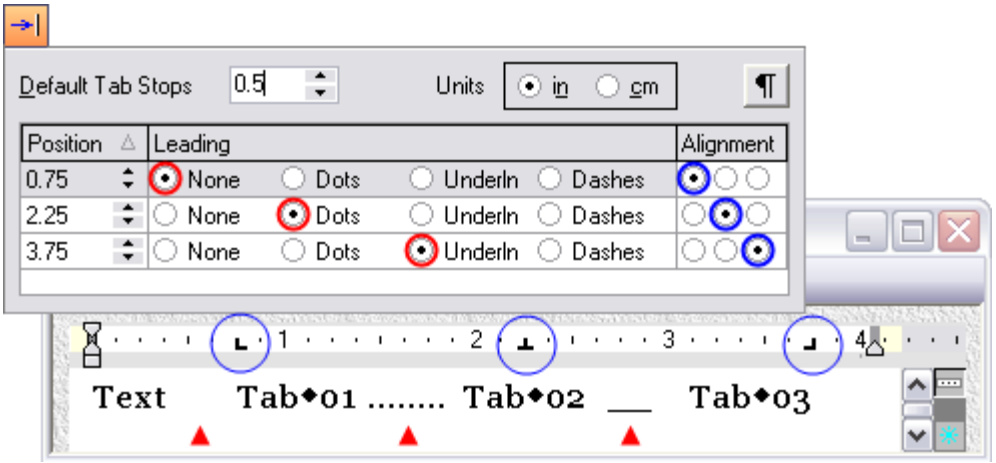
The Units radio group allows you to change the current Memo's units to either inches or centimeters. This will only affect the current Memo, and will only remain in effect until its InfoView is closed. To make all newly-opened Memos default to a particular unit, you'd set your preference in the Settings form.

The toggle button with an End-of-Paragraph mark in the top/right corner is the Show Special Symbols button. Depress it to make the current Memo display Tabs, Spaces, and End-of-Paragraph marks. Can be useful while playing with Tabs.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



We now get to the most interesting part of the Tabs dialog: the Tab List. This is where all user-defined Tab stops are listed. You can manipulate existing Tabs by changing their Position, Leading (what line (none, dotted, underline, dashed) leads to the text aligned at this tab stop), and Alignment (Left/Center/Right – how the text is positioned relative to this tab stop). You can delete existing Tabs by deleting the corresponding record in the list (just hit Delete or Ctrl-Delete). You can create new Tabs by inserting new records in the list (hit Insert when not inside a cell of an existing Tab record, if any. Hit Enter to get out of the cell if necessary).



2.6.10 Memo Tables

Alventis Memos support complex nested tables. We will only be talking about Memo Tables in this chapter. Tables should be familiar to you from the most common word processors. Little, if anything, is different in our Memos. Nonetheless, let's start by pointing out some table features you may find interesting.

Tables are nested or hierarchical. This means that you can have a table "sitting" in a cell of another table. Generally speaking, each cell of each table can contain whatever any Memo can contain: text, pictures, more tables, and so on. There is no pre-set limit on the nesting level, i.e., you can have a table in a cell of a table, which is in a cell of a table, which is... you get the idea.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Technically-speaking, tables resemble HTML-style tables more than, e.g., ones found in Microsoft Word 97. That is, each cell has 4 border edges + the table itself has same. This contrasts with older-style tables that were comprised of lines between cells, and where cells did not have a concept of borders "belonging" to them.

Tables can have either a fixed width set by dragging the right edge of the table with the mouse – or a width that is set as a percentage of the width of the Memo.

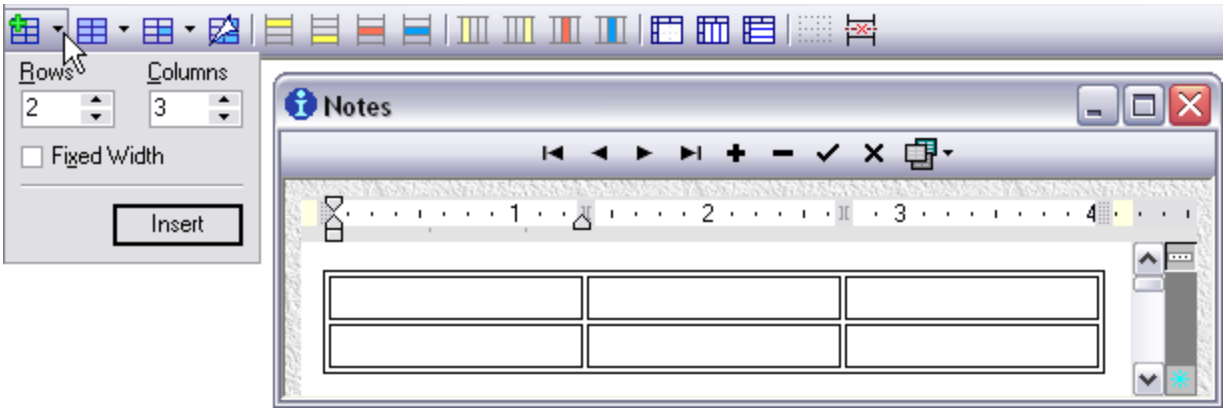
A table is considered selected under the following circumstances: it is really selected by clicking with the mouse in the margin just to the left of the table; the cursor is in a cell of this table; and finally, one or more cells of this table are selected.

Cells are selected by dragging the mouse from one cell to another. You can also select cells using the Select Rows or Select Columns commands. If you want to perform some operation on a single cell though, you don't need to select it at all: just placing the cursor in that cell is enough.

As with many word processors, you navigate between cells by hitting the Tab key (or Shift-Tab to go in the opposite direction), you can insert Tab characters in a cell by hitting Ctrl-Tab, and you can extend the table one row at a time by hitting Tab in the last cell. You can resize columns and rows by dragging their edges.

But enough about what we *can* do, let's do something. Let's start by learning to create tables.

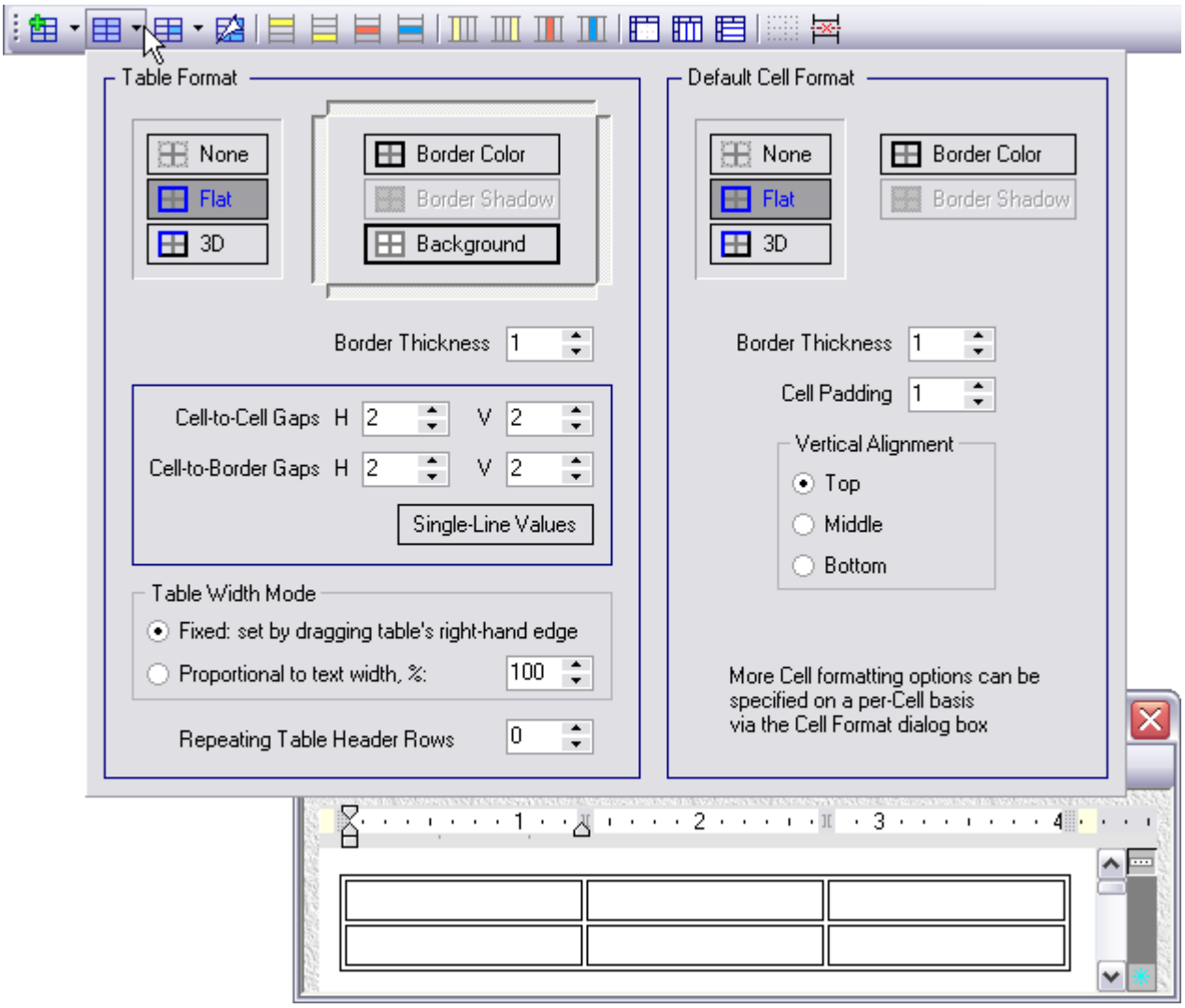
You create new tables using the Insert Table InstaButton. Being an InstaButton, it follows the principles of operation of all InstaButtons.



Insert Table dropdown is the smallest dialog in Alventis. You specify the dimensions of the table you want to create (how many Rows by how many Columns). You also specify whether you want a Fixed-width table (check the checkbox) or one whose width is proportional to the width of the Memo. We'll talk about this in more detail in a bit, when we get to the Table Format dialog. Once satisfied with the initial settings for the table, you can hit the Insert button to insert the table at the current cursor position in the Memo. From now on, this InstaButton remembers what kind of table it is set-up to create, so you can create more tables by simply clicking on the button. Now that we have our table, it's time to improve its appearance.

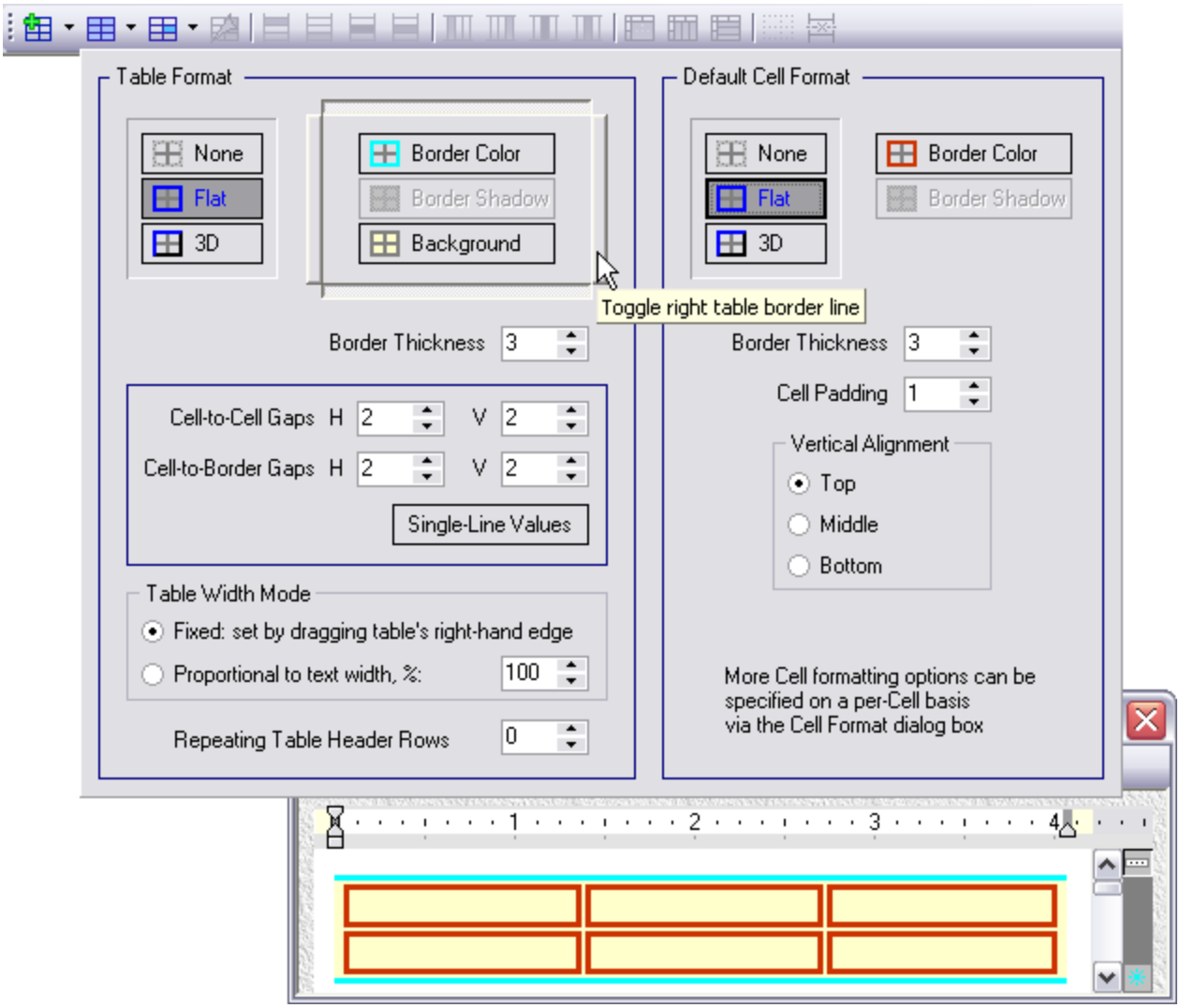
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The Table Format InstaButton is responsible for the overall appearance of the table. Let's see what its dropdown dialog looks like.



The dialog is divided in two areas. On the left, you find settings that apply to the entire table as a whole. On the right are settings applicable to cells. These are merely default settings for cells: each cell may override some of them on an individual, per-cell basis. We'll get to this subject later, when we talk about the Cell Format InstaButton, but keep in mind that there's more to cell format than meets the eye in this dialog.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

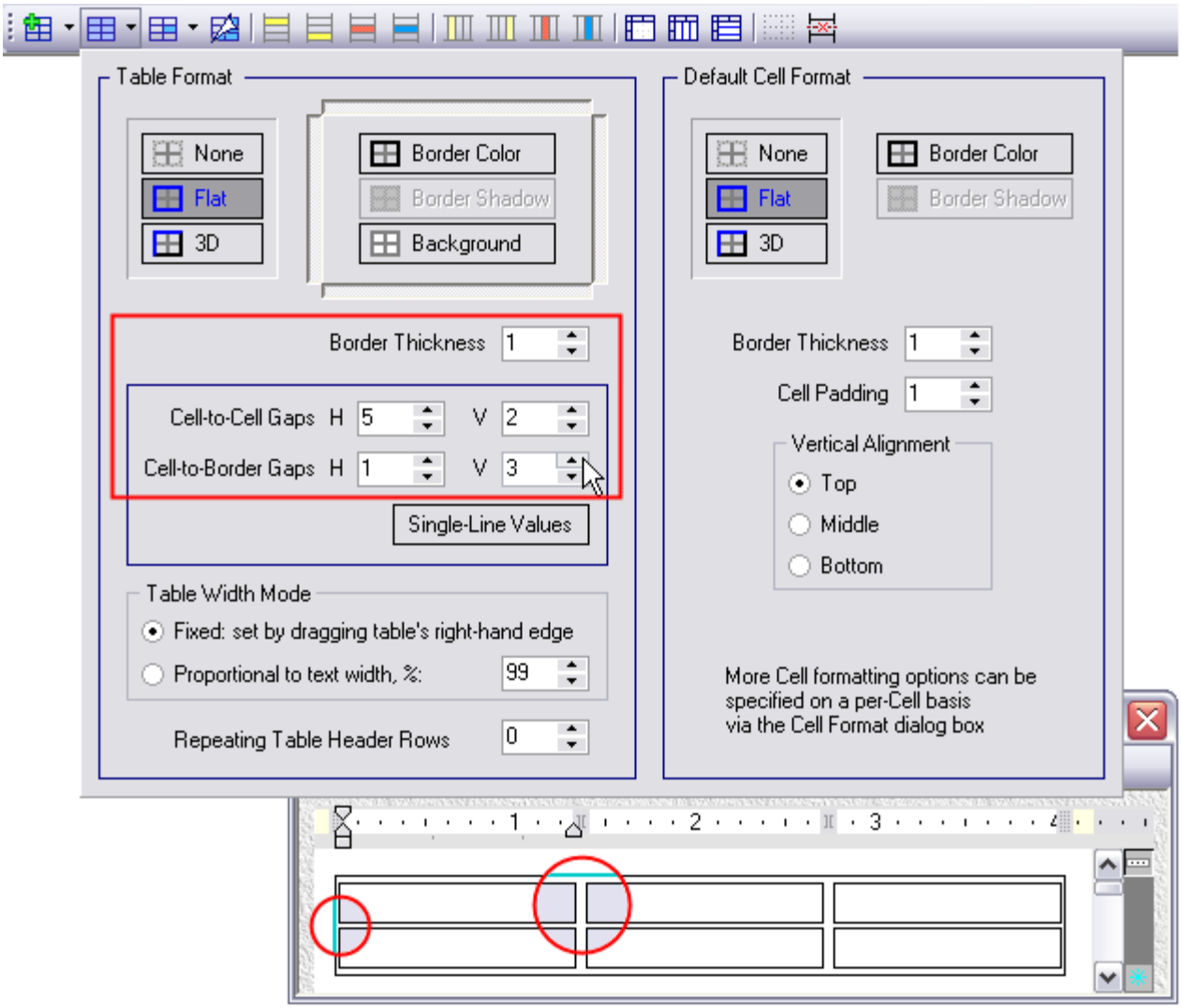


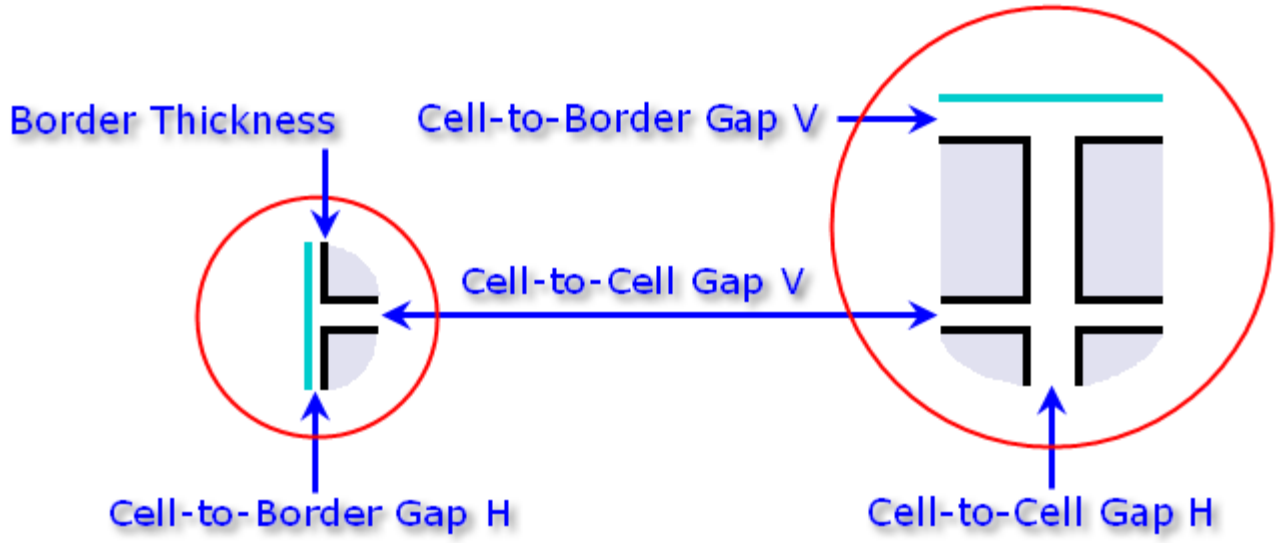
Back in the Table Format (left-side) area of the dialog we find a group of controls responsible for the appearance of table borders and background. Once again, table borders are just the 4 edges of the big "box" that has cells in it – and not all lines that comprise the table. This "box" can have no border at all (None), a Flat border drawn with a single Border Color, or a 3D border drawn with two colors: Border Color and Border Shadow color (no, it's not a real shadow, just the color of the right and bottom edges of the table). A table can have a Background color. Set it to "None" to make the table transparent, i.e., let the background color of its paragraph, if any, show through. You can make each edge of the table drawn or not drawn by toggling the corresponding edge of the rectangle that surrounds the above-mentioned buttons.

Finally, you can change the thickness of the border using the SpinEdit box named accordingly.

The next group of controls has to do with gaps between cells. Cell-to-Cell Gaps are distances (in pixels) between adjacent cells. The "H" gap applies to left-right neighbors and the "V" gap – ones that are adjacent vertically.
Cell-to-Border Gaps are similar, but apply to distances between cells and borders of the table itself.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.





Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The Single-Line Values buttons performs a simple trick which exists for your convenience. What it does is set things up in such a way that tables start to look like those from the Word 97 era: single lines everywhere. Specifically, it sets Border Thickness (of both the table and the cells) to 1, and all Gaps to -1, which makes borders of adjacent cells overlap, which makes it look like there's only a single line between the cells.

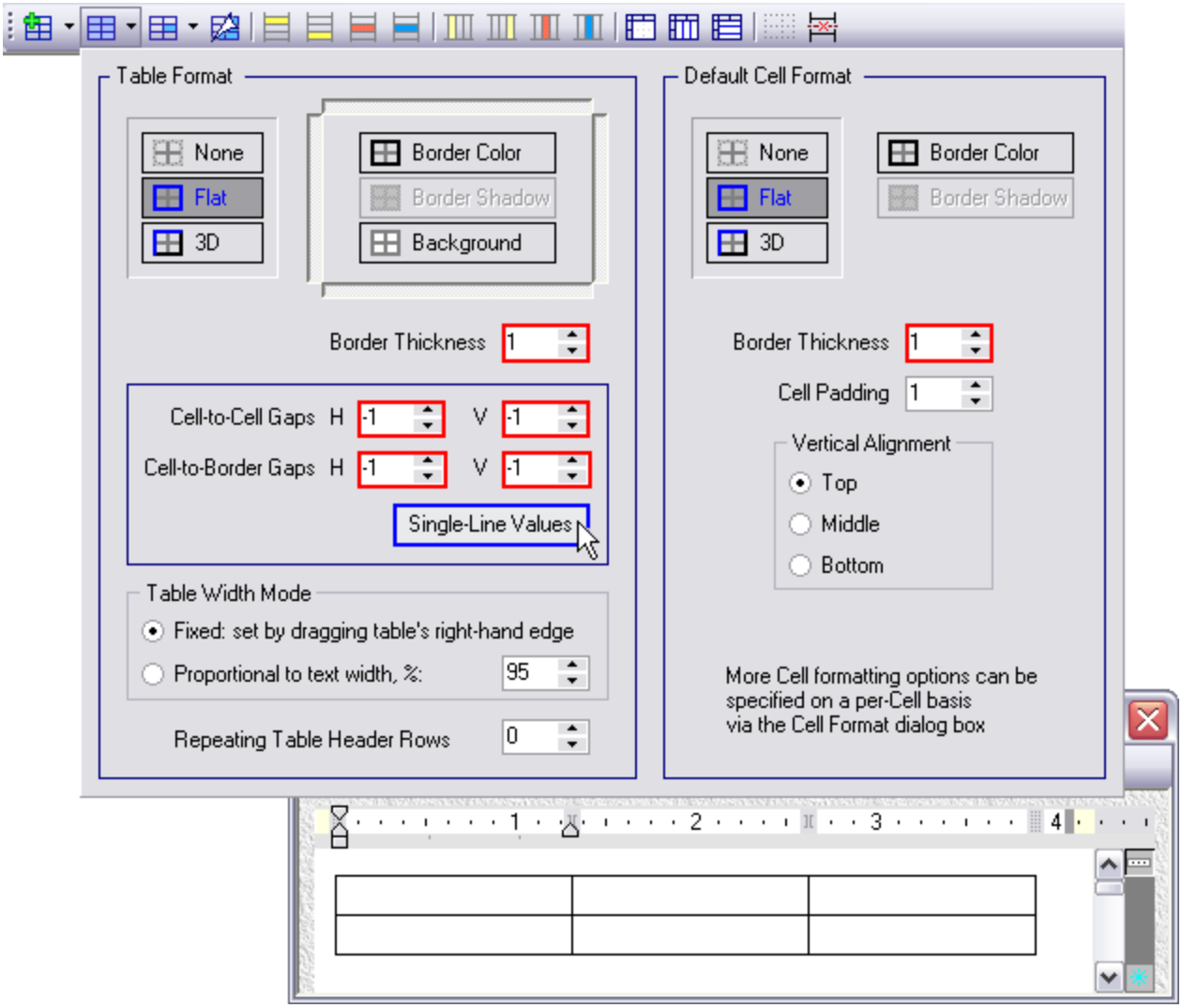


Table Width Mode is the next and very important radio group. It pretty much spells out what it does, but let's go over this in a bit more detail.

Fixed-width tables maintain a certain width that initially (when the table is first created) equals that of the Memo, and can be set at any time by dragging the right edge of the table left or right with the mouse. Note that there is no way to set it numerically. When you resize the Memo, the table "stays put" and maintains its fixed width.

Proportional tables always maintain their width relative to the width of the Memo. A Proportional table with a 100% width (set in the corresponding SpinEdit box) will resize itself to always occupy the entire width of the Memo. You can't resize such a table with the mouse. An attempt to drag its right edge will affect the widths of its columns though, so don't try it unless you want the columns to get resized

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

inversely-proportionally to your table-resizing attempt. If this is not clear, try it an see for yourself.

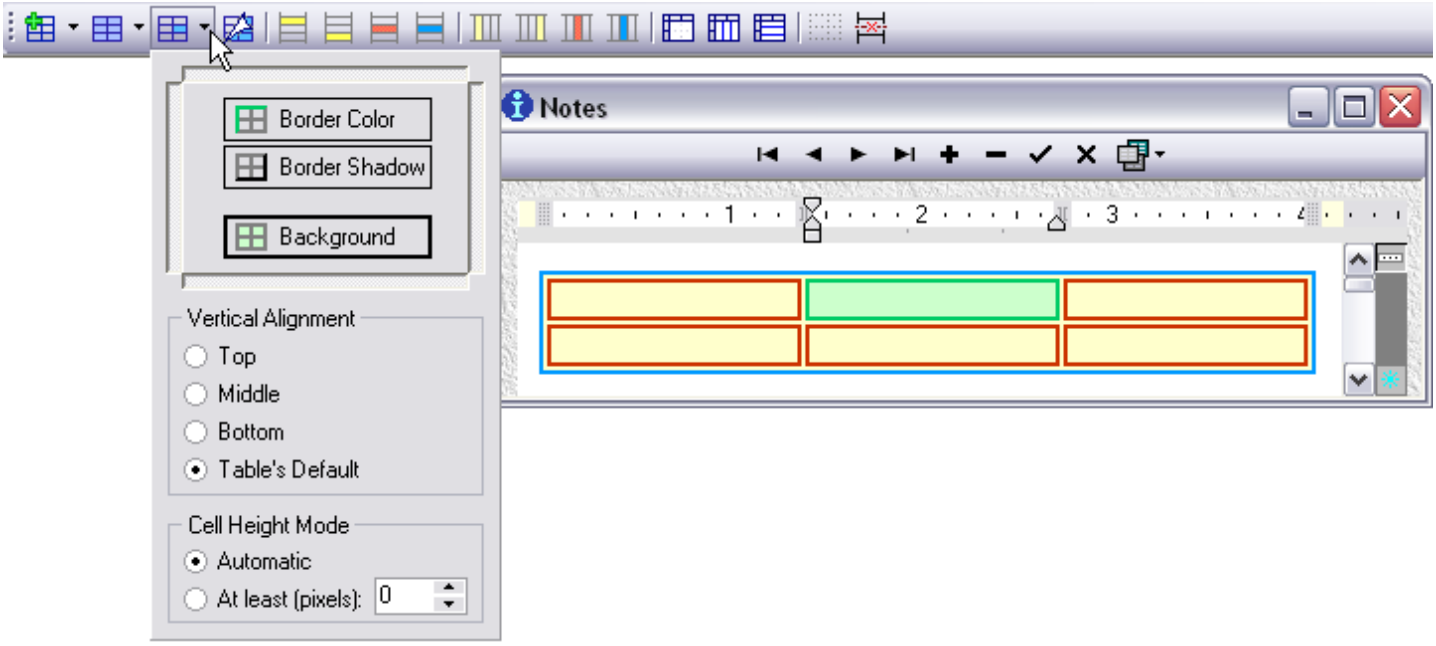
At the very bottom of this section of the dialog we find the Repeating Table Header Rows SpinEdit box. This attribute is only applicable to printing, so don't expect to see any immediate effect. What it specifies is how many rows of the table are to be considered Header Rows and should therefore be printed repeatedly at the top of every page that ends up with a portion of this table.

The Default Cell Format section on the right-hand side specifies some default formatting attributes applicable to cells. The first group of controls does exactly the same thing their counterparts on the left do to the table: None/Flat/3D specify overall border style, Border Color/Shadow specify one or two colors of the border, and Border Thickness specifies just that.

Cell Padding is something new though. This specifies (in pixels) an additional amount of space you would like to leave between the cell's edges and its contents.

The Vertical Alignment radio group sets the default alignment for cell contents: Top, Middle, or Bottom. This will only come into play if you end up with a cell that is taller than its contents.

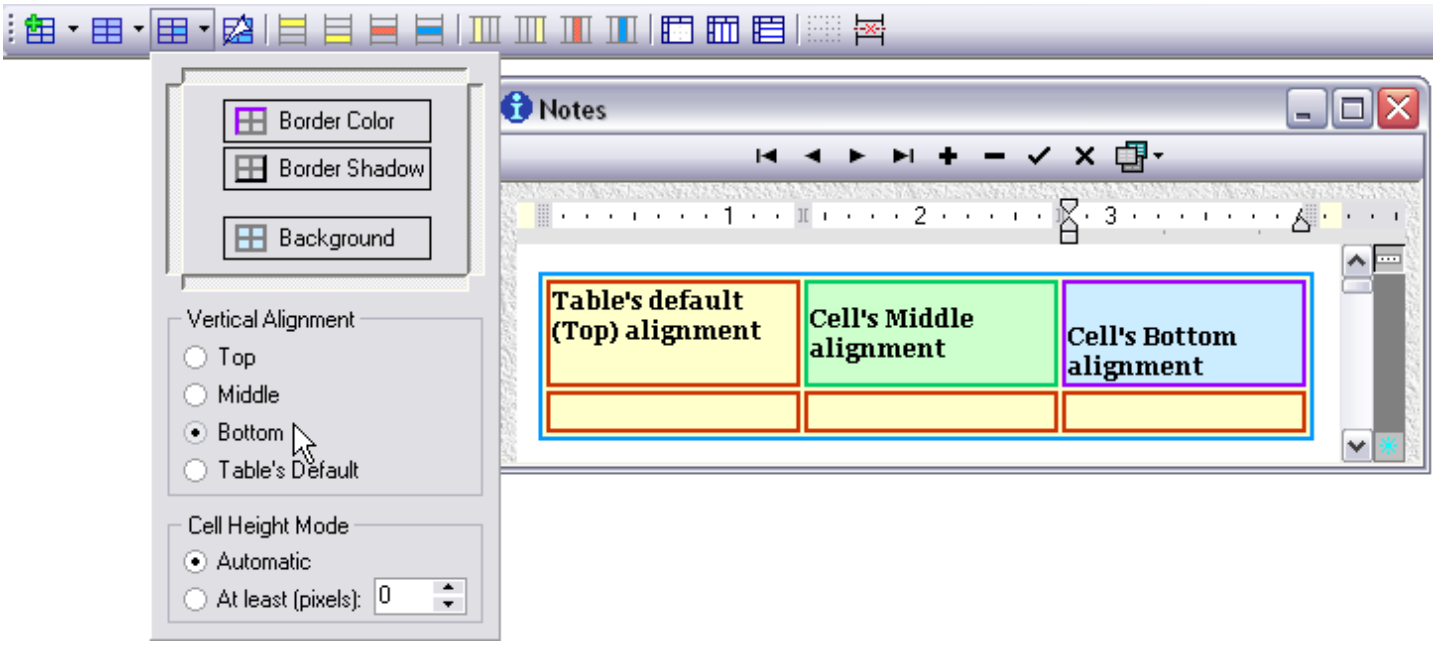
The Cell Format InstaButton is the last InstaButton relevant to table/cell formatting, and luckily it is rather simple. Everything here applies to individual cells of a table.



At the top we yet again find the now familiar Border/Background control group. It has been described in detail at the beginning of the Table Format topic, so we won't repeat its rather obvious workings here.

The Vertical Alignment radio group has also been dealt with earlier. This group is very similar with one notable addition: the Table's Default option. This setting tells the cell to simply use the default Alignment specified for the entire table in the Table Format dialog. The other 3 options tell the cell to override the table's default and use the one you select here.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Cell Height Mode group controls the height of the cell. Automatic mode is the default and means the cell has a minimum height of a single line of text and grows as necessary to accommodate its contents. Alternatively, you can specify a minimum height of a cell in pixels. It will not shrink smaller than that, but will still grow if necessary.

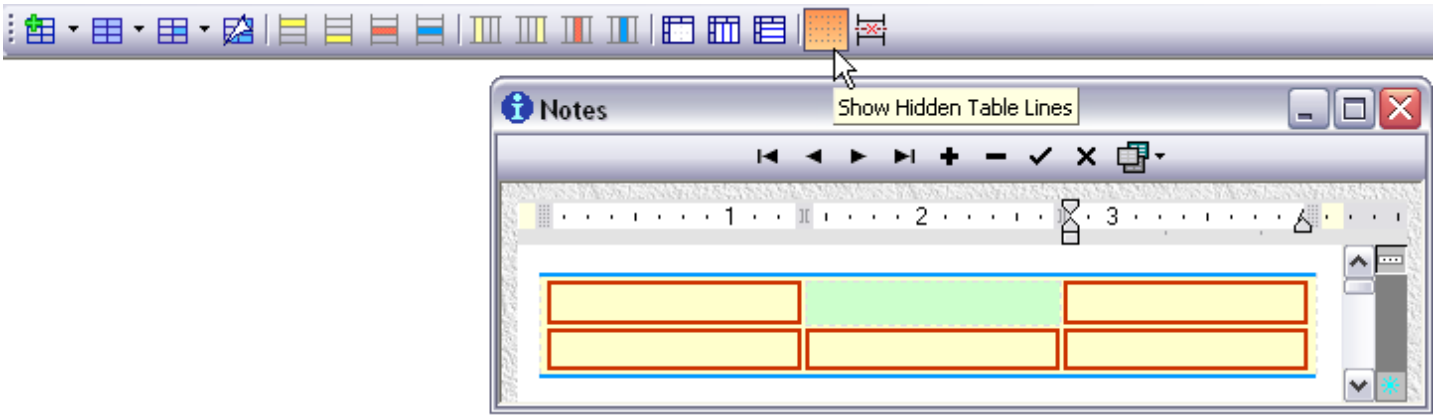
The Clear Cell Formatting button allows you to quickly return the cell to its defaults, so its format becomes once again specified by the Table Format / Default Cell Format.

Keep Table Together is a toggle button. When depressed, it tells the table to try to not span multiple pages when getting printed. This is akin to the similarly named Paragraph Format option. Alventis will attempt to print the table on a single page if it fits.

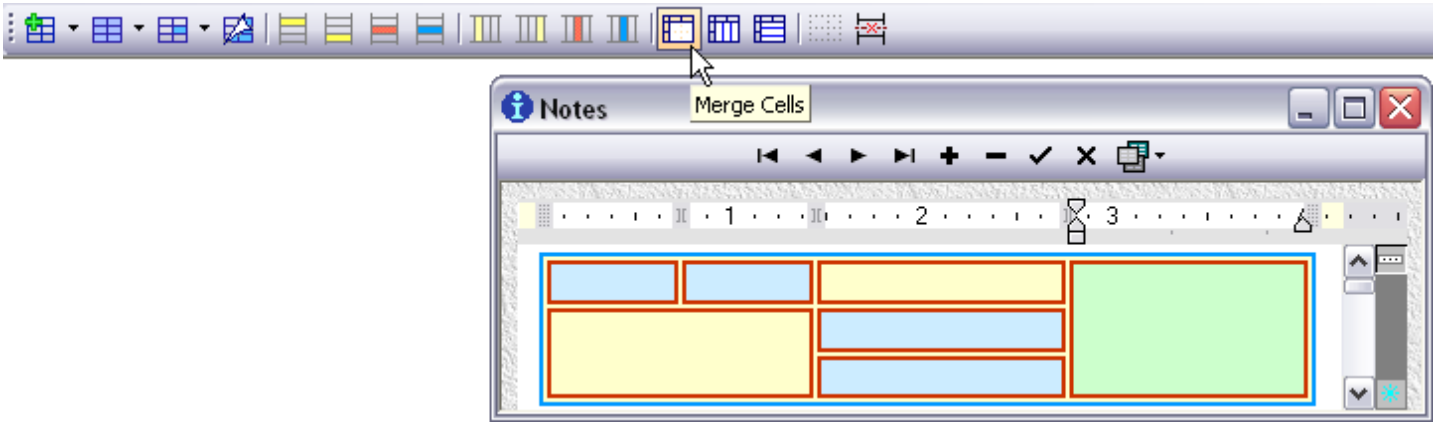
While we're on the subject of printing, it's only fitting (pun intended) to mention a limitation you should be aware of. If you create a cell so tall that it doesn't fit on a single page, its bottom will not get printed. It will simply get chopped-off when it reaches the end of the page. So don't create huge cells if you intend to print the Memo that contains them.

Show Hidden Table Lines is another toggle button. Depressing it makes the table's hidden lines (if any) displayed. Remember all those table and cell borders you could turn On and Off? A "hidden line" is a dotted line that, when this button is On, indicates the positions of the disabled borders.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Memo tables support Cell Merging and Splitting.



The Merge Cells button merges the selected cells together, if possible.

Split Cells Vertically splits the cell (or indeed the selected cells) in two in a side-by-side fashion.

Split Cells Horizontally does the same but in the other direction.

Finally, there also exist a bunch of buttons so simple they barely merit mention:

Insert Row Above inserts a new row above the selected cell(s).

Insert Row Below – same but below.

Insert Column on the Left creates a new column to the left of the selected cell(s).

Insert Column on the Right – same but on the right.

Delete Selected Rows deletes all rows containing the selected cell(s). Note that you don't have to select an entire row to delete it.

Delete Selected Columns – we'll leave it for you to puzzle out.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

2.6.11 Pictures

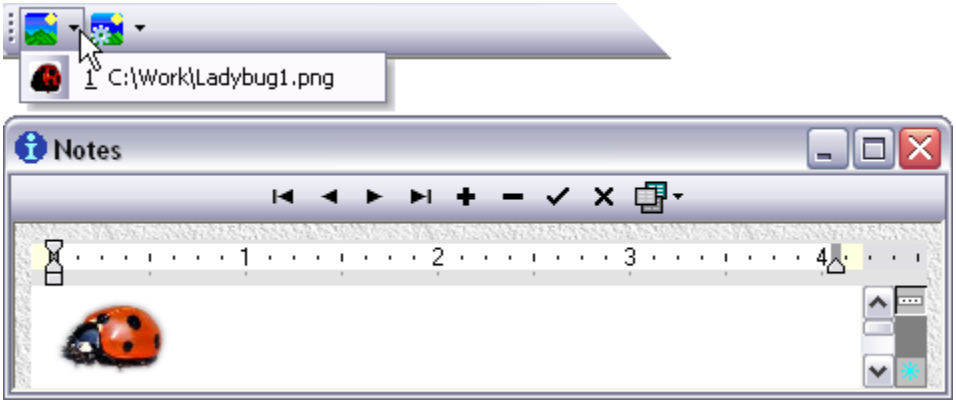
You can insert pictures in Memos. There's one command to insert a picture from a file and another – to adjust the formatting of that picture .

Alventis supports the following image formats: Bitmaps (BMP), JPEGs (JPG/JPEG), GIFs (GIF), PNGs (PNG), Icons (ICO), Enhanced and regular Metafiles (EMF/WMF).

Support for GIF images includes support for transparency and animation. To toggle animation On/Off for all Memos use the Enable GIF Animation button .
PNG images can also be transparent, and this too is supported.

Inserting a picture is easy. Click the Insert Picture button, select the picture you want from the standard Open dialog, and you're done.

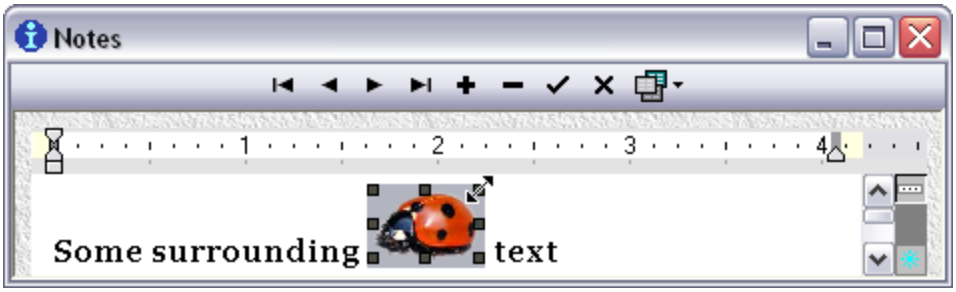
If you ever need to re-insert the same picture again, it's even easier: just drop-down the list of most-recently inserted pictures and pick the one you want. The list is accessible by clicking on the little arrow at the right edge of the very same Insert Picture button.



Once we have our picture in the Memo, we can do a few things with it (aside from admiring its beauty). We can click on it to select it or do the same by selecting it with the mouse or via the keyboard as if it were a single character. This actually pretty much describes how pictures are aligned/treated in Memos: like a character, perhaps a very large one, but character nonetheless.

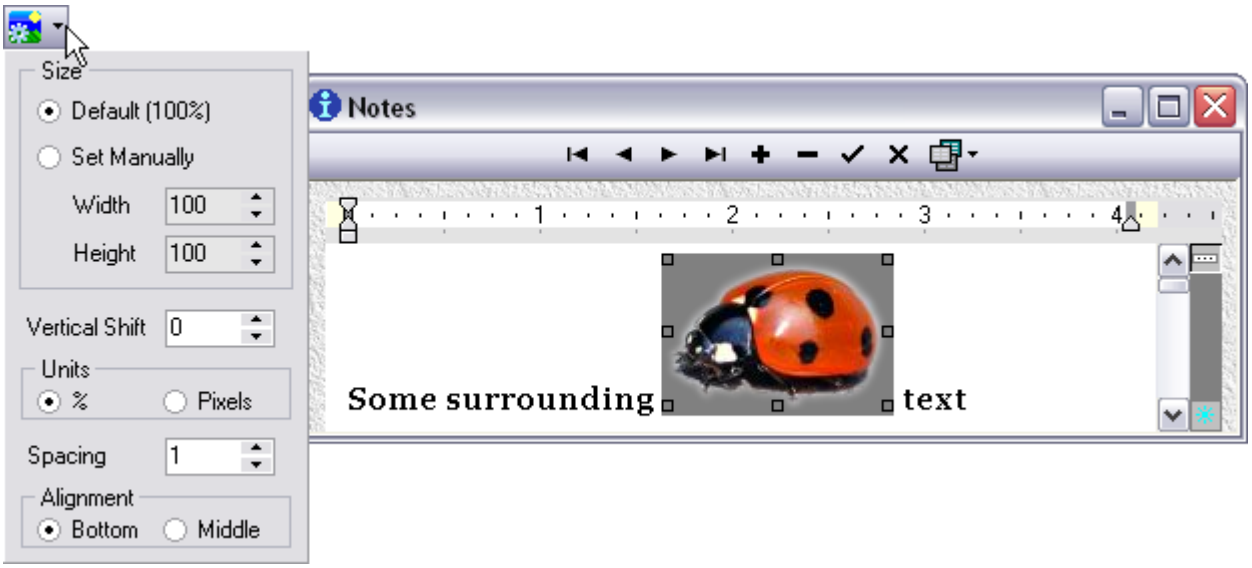
When the picture is selected, little sizing handles appear in its corners and at the middles of its sides. You can resize the picture with the mouse by dragging these handles around. Resizing by the corner always resizes it proportionally, i.e., it preserves the picture's aspect ratio. Resizing by the side handle allows you to go wild and freely change the aspect ratio.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



You can also drag the selected picture wherever you want in the same (or another) Memo. Drag-and-drop is a basic enough technique, so we won't go into it. Suffice it to say that you click the picture, drag the mouse where you want the picture to end up, and release the mouse.

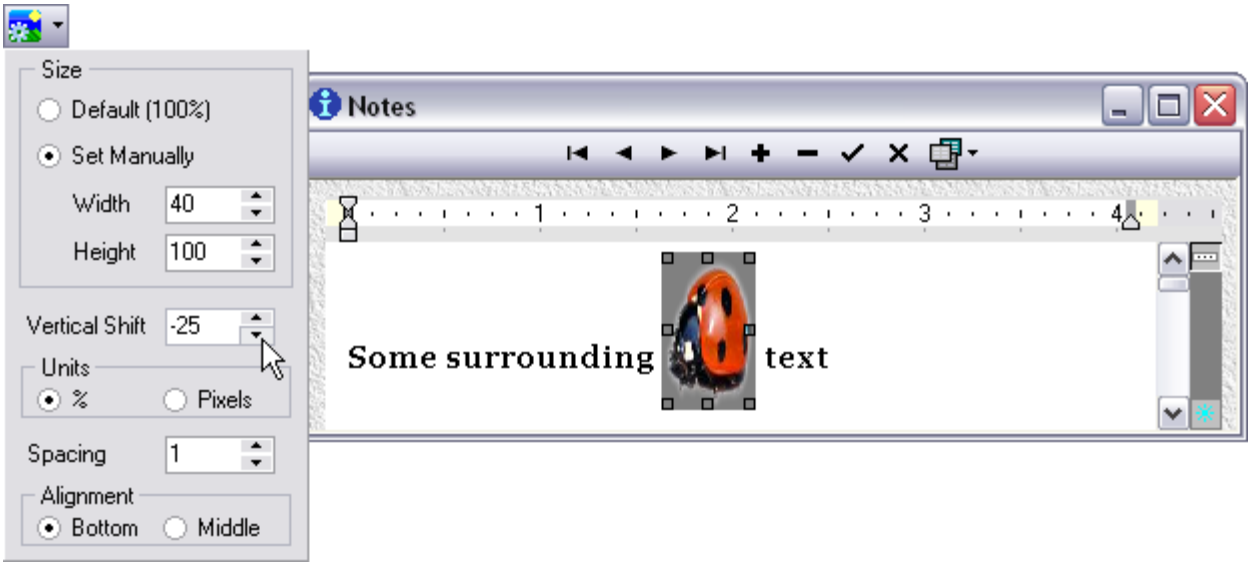
There are a few other things you can adjust about the picture, but they require a dialog box.



The Picture Format is an InstaButton, so it follows the principles of operation of all InstaButtons. Its dropdown dialog allows us to specify the picture's size. We can either set it to Default, which means the picture's original size or 100%. We can also set its dimensions manually as either a percentage of its original size or explicitly, in pixels – depending on the selected Units (see below).

Vertical Shift specifies how much the picture should be shifted relative to its normal alignment. Positive values shift it upwards, negative – downwards.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

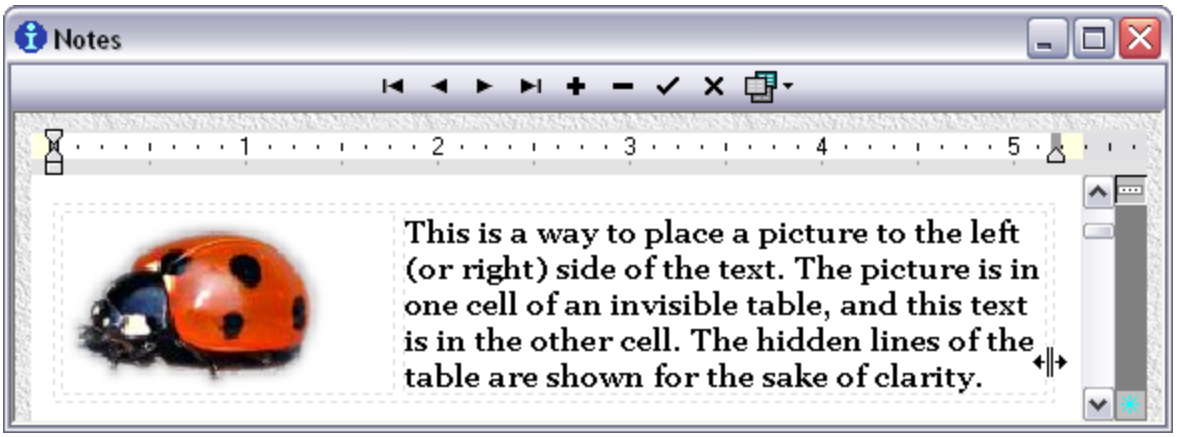


Units specify how you want to measure the Size and Vertical Shift: as a percentage of its original size or in pixels.

Spacing sets the amount of blank space you want to leave between the picture and its neighbors (text or other pictures).

Alignment selects how you want to align the picture relative to the baseline of the text on the same line. Bottom makes the picture "sit" on top of the baseline, much like an uppercase letter would. Middle aligns the mid-point of the picture with the baseline. Don't forget that you can still nudge the picture up or down by adjusting the Vertical Shift.

At this point you may be wondering how you can place a picture on the left and make the text flow around it on the right. The truth of the matter is: you can't. At least not directly. Floating pictures or similar picture alignments are not supported. The only way to arrive to the layout you may be after is to use Tables. For example, you could insert a suitably-sized table in a Memo (a single row with 2 columns would likely suffice). You would then insert the picture in the left cell and put the text in the right one. Depending on a variety of factors this may or may not prove to be particularly difficult.


Notes
This is a way to place a picture to the left (or right) side of the text. The picture is in one cell of an invisible table, and this text is in the other cell. The hidden lines of the table are shown for the sake of clarity.

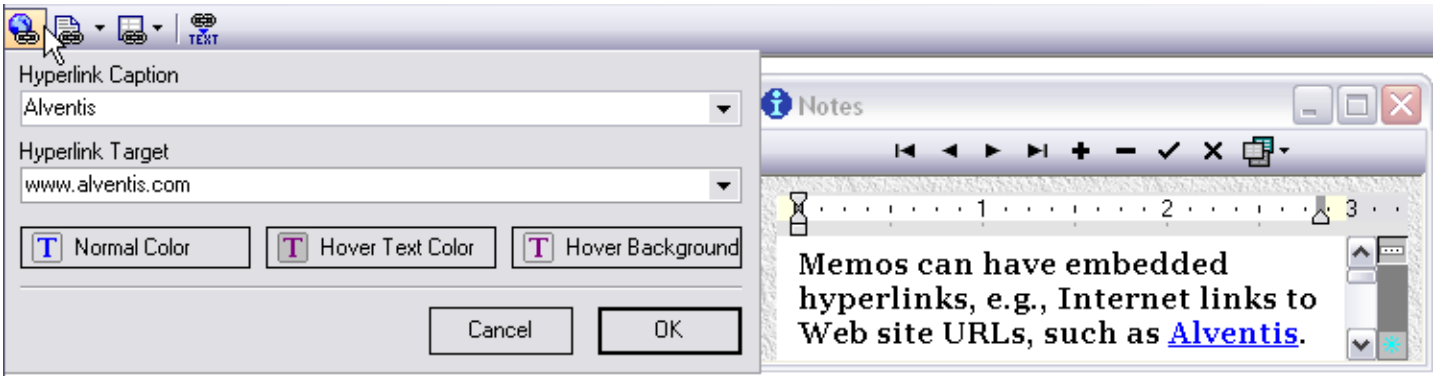
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

2.6.12 Hyperlinks

Alventis Memos support **Hyperlinks** of several flavors. On the whole, the way they work is very similar to what is commonly found in other word processors. Invoking a Hyperlink may be a bit different though. When a Memo is in a saved state, clicking on the link with the mouse is sufficient. In a modified Memo, you must hold down the Ctrl key while clicking the link to activate it. Regular clicking simply places the cursor at the point you click.

Insert/Edit Hyperlink allows you to create and edit Hyperlinks. If you invoke this command with the cursor on an existing Hyperlink, you will get a chance to edit it. If the cursor is not on an existing link, you will have an opportunity to create a new Hyperlink from scratch. The only difference between the above two scenarios is whether you are modifying an existing link or creating a new one – the interface and the procedure are exactly the same, so we'll discuss both at the same time.

Clicking on the aforesaid button opens a dropdown dialog that looks like this:



Note that while the dialog's behavior resembles that of InstaButton dialogs (it gets dismissed if you click outside of it), the Insert/Edit Hyperlink button is not an InstaButton.

The Hyperlink Caption MRU Combo Box allows you to edit or create a Caption for your Hyperlink, i.e., the text that will appear in the Memo.
The Hyperlink Target MRU Combo Box allows you to specify the actual link target. More on that later.
Let's finish with the few remaining controls first.
The 3 buttons under the combo boxes select the colors you'd like your Hyperlink's Caption to have.
Normal Color is the color of the Hyperlink that just "sits quietly" in its Memo.
Hover Text Color and Hover Background are the colors the Hyperlink is painted with when you pass the mouse cursor over the link or place the editing caret in it. This is very much like it works in your favorite Web browser.
Note that you can change many of the link's font attributes outside of this dialog (we'll tell you how in a bit) but it is only here that you can specify its Normal and the two Hover colors.

Hyperlink Targets.

While we're still in the Hyperlink dialog, let's discuss what sort of things you can create links to. But first, the good news: you can create Hyperlinks by hand as much as you want, but there are several easier ways to create them using several simple command buttons. We'll talk about them later.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

URL Hyperlinks. These would be the Web links. Some examples are:
www.alventis.com
http://www.alventis.com
http://www.google.com/search?q=Alventis
mailto:someone@somewhere.com
...and whatever else can be interpreted by your default Web browser.

File Hyperlinks. You can create links to files on your computer.

The Target will look something like this:
C:\Path\FileName.txt
"C:\Program Files\Another\File.xyz"

Note that file names that include spaces must be enclosed in double quotation marks. The filenames must be specified in a way that will be understood by the Operating System if it tries to open that file from the Alventis application directory. This is guru territory though, and you should normally specify the full filename with the full path – as in the above examples.

You can insert links to files by Alt-dropping a file from, e.g., Windows Explorer: start dragging a file as usual, but make sure you hold down the Alt key when dropping the file in the Memo. If you don't hold down the Alt key, Alventis will attempt to insert the *contents* of the file into the Memo - if it is of a recognized type (TXT, RTF, Pictures).

If you are really brave, there's a special expert feature for you. Please note that it is relatively advanced and assumes that you know how to execute programs from the command line. If this is not the case, you can safely skip this section and go straight to the one on Record Hyperlinks.

You can specify not merely document-type files (TXT, DOC, various image files, to name a few possibilities) as Targets, but applications too. For instance, you could set the Target to:
Calc
That's right, this would be enough to open the Windows Calculator directly from a Memo. You don't have to specify the .exe extension in this case: Windows assumes it by default. That's nice, but what if you were even braver and wanted to launch a program with command-line arguments? No problem: just enter the link Target as you would enter the command in the Run dialog. If the executable contains spaces, you must enclose it in double quotes. Technically, everything after the first space which is not within the enquoted executable's name, is treated as the parameter string. The following are some valid examples:
Notepad C:\SomePath\FileName.txt
Notepad "C:\Path with spaces\File.txt"
"C:\Some path with spaces\Program.exe" param1 param2

Record Hyperlinks. Last but not least, you can have links to records within Alventis itself.
The format for such links is:
intralink://Zero.DBID.TableID.RecID

Zero is just that: number 0, always the same.
DBID is the ID number of the Database.
TableID is the ID of the Table.
RecID is the ID of the Record in that Table.

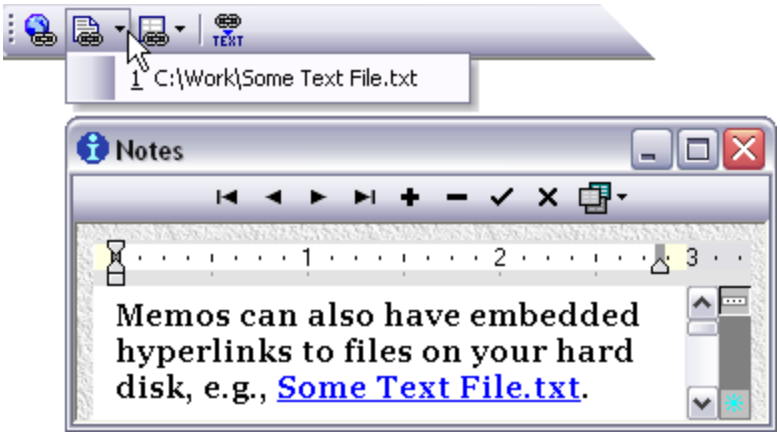
So, if you wanted to create (by hand) a Hyperlink to Record 23 of Table 5 of Database 2, the link Target

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

would look like this:
intralink://0.2.5.23

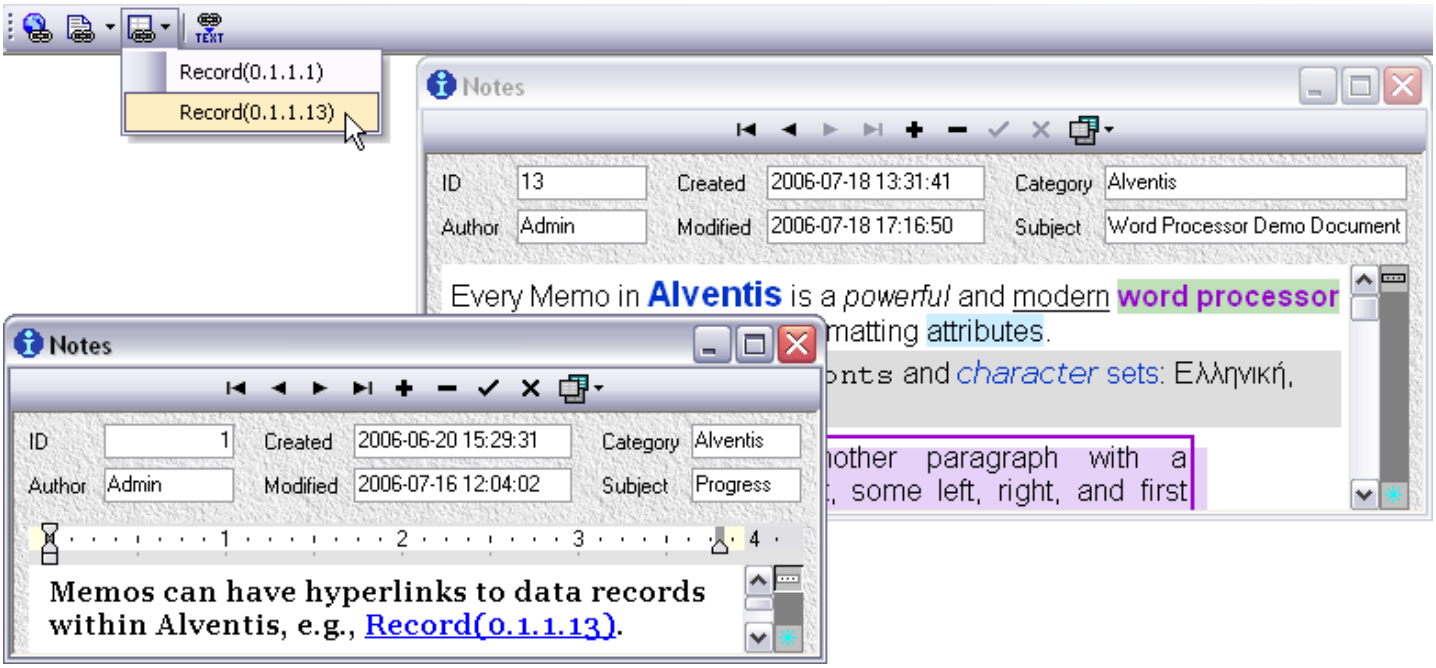
As promised, let's see now how we can create some of the above Hyperlinks without tediously typing their Targets by hand.

Insert Link to File button does precisely that: it allows you to simply pick a file from the standard Open dialog box – and the link is created for you. The button remembers the most recently inserted links, so you can re-insert a link by simply picking it from the little menu that opens when you click the little arrow on the right-hand side of the button.

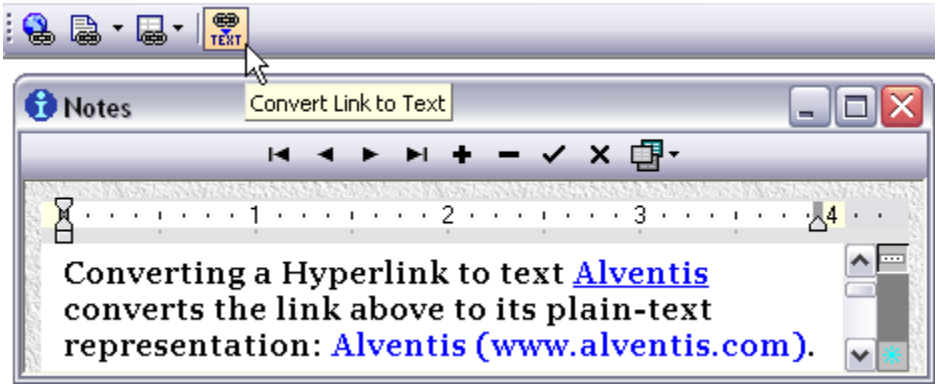


Insert Link to Record drops down a menu with a list of all currently open records. Convenient if you want to link to another record open in another InfoView.
If the record you want to link to is not currently open but appears in the Search Results in a UniGrid, then all you need to do is drag-and-drop that record into the Memo.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Convert Link to Text button does not insert a link at all but let's get it out of the way. It simply takes all Hyperlinks within the selected text (or absent that, the one the cursor is on), and converts them to plain text of the form Caption (Target). This may be useful if, e.g., you want to print some text that contains Hyperlinks and you want to see the actual Targets and not just their Captions in the printout.


Notes
Convert Link to Text
Converting a Hyperlink to text Alventis converts the link above to its plain-text representation: Alventis (www.alventis.com).

Inserting Web links is not too difficult either. In Microsoft Internet Explorer, for example, select one or more Hyperlinks and copy-and-paste the selection into a Memo. You can also use drag-and-drop. In Mozilla Firefox, copy-and-paste inserts merely the text, so you lose all link Targets. Drag-and-drop of a single link does work though. Unfortunately, Opera doesn't help you with the task of Hyperlink insertion: drag-and-drop doesn't work at all, and copy-and-paste only gives us the text. So, if Opera is your favorite, you'll have to resort to inserting Hyperlinks manually: right-click on the link and select Copy link address from the popup menu, and then use the Insert/Edit Hyperlink button to insert it into the Memo.

You can paste the HTML-formatted contents of the clipboard into the Memo using the Paste in Memo as

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

HTML command . Note that if the clipboard does not contain any HTML, the command will do nothing - even if the clipboard contains some other recognizable formats (e.g., plain text or RTF).

You can also try to import the entire HTML file into the Memo, but this is a subject of an entirely different chapter called Memo Import/Export.

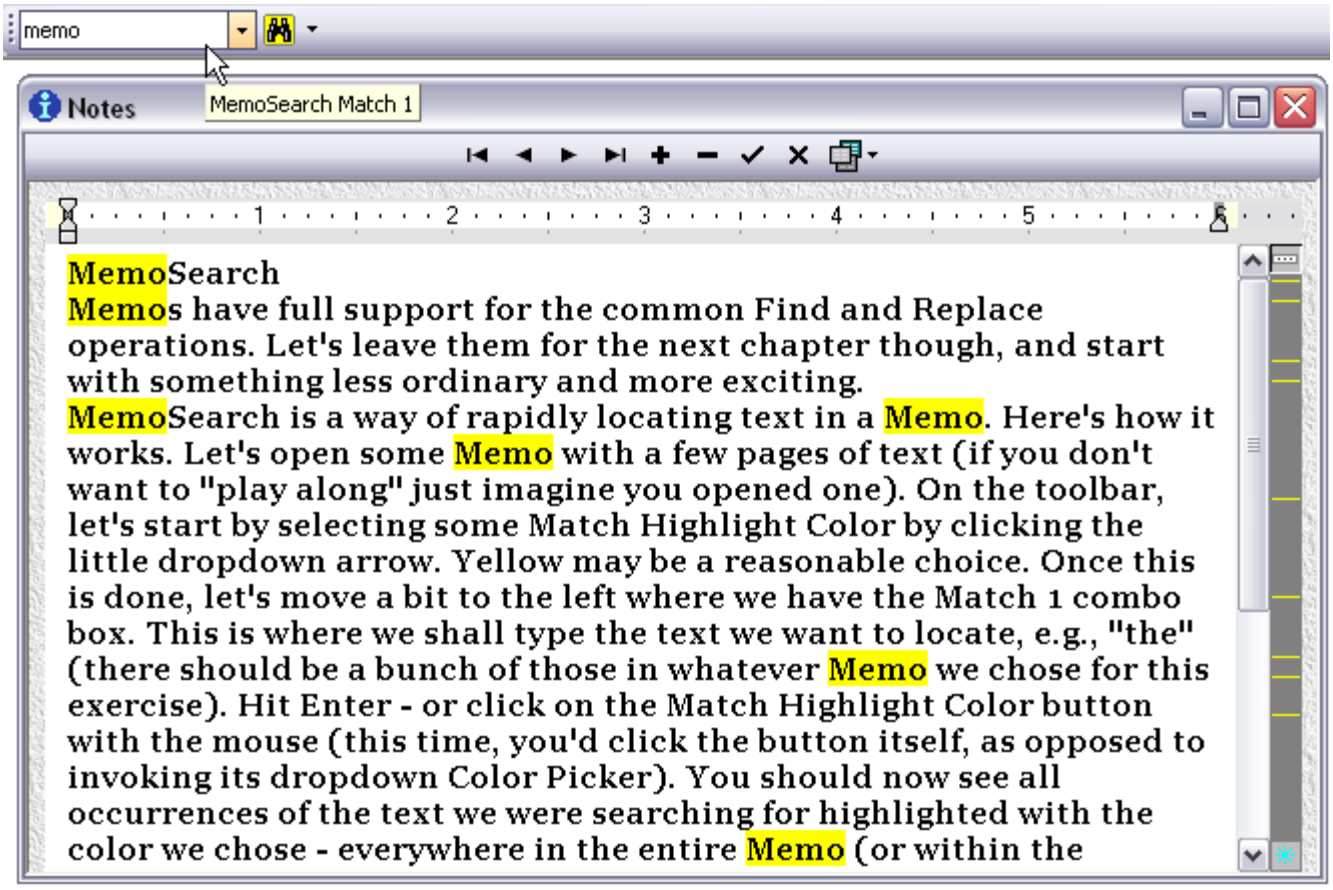
Hyperlink Formatting. You can change the Hyperlink's colors via the Insert/Edit Hyperlink dialog (which is the only place where you can set them), but you can adjust the link's appearance directly as well. Simply place the cursor anywhere within the Hyperlink (or select any part of it) and use all the usual formatting tools: Font, QuickStyle, etc.

2.6.13 MemoSearch

Memos have full support for the common Find and Replace operations. Let's leave them for the next chapter though, and start with something less ordinary and more exciting.

MemoSearch is a way of rapidly locating text in a Memo. Here's how it works. Let's open some Memo with a few pages of text (if you don't want to "play along" just imagine you opened one). On the toolbar, let's start by selecting some Match Highlight Color by clicking the little dropdown arrow. Yellow may be a reasonable choice. Once this is done, let's move a bit to the left where we have the Match 1 combo box. This is where we shall type the text we want to locate, e.g., "the" (there should be a bunch of those in whatever Memo we chose for this exercise). Hit Enter – or click on the Match Highlight Color button with the mouse (this time, you'd click the button itself, as opposed to invoking its dropdown Color Picker).

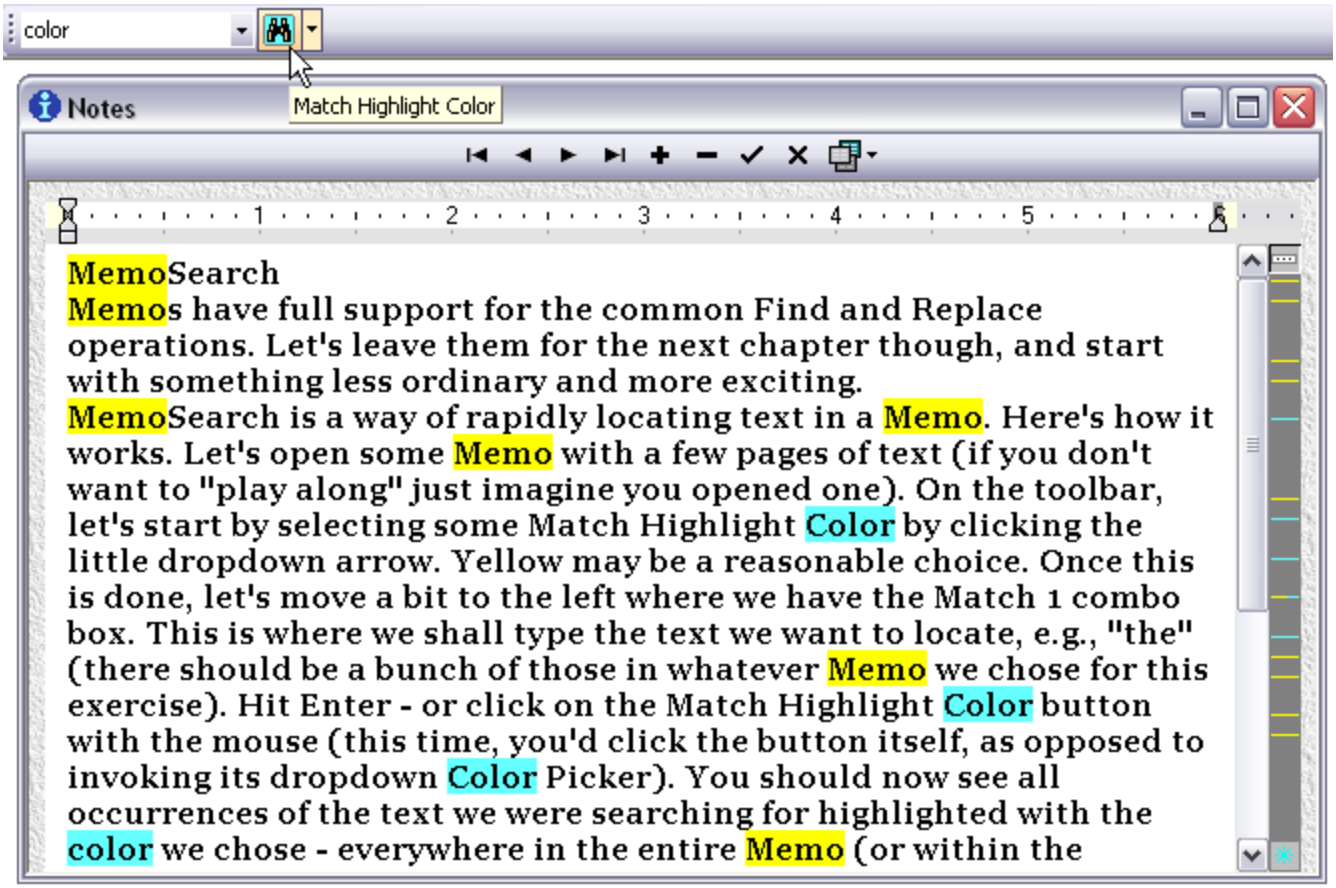
Copyright © 2002-2007 Alventis Corporation. All rights reserved.


memo
Notes
MemoSearch Match 1
MemoSearch
Memos have full support for the common Find and Replace operations. Let's leave them for the next chapter though, and start with something less ordinary and more exciting.
MemoSearch is a way of rapidly locating text in a Memo. Here's how it works. Let's open some Memo with a few pages of text (if you don't want to "play along" just imagine you opened one). On the toolbar, let's start by selecting some Match Highlight Color by clicking the little dropdown arrow. Yellow may be a reasonable choice. Once this is done, let's move a bit to the left where we have the Match 1 combo box. This is where we shall type the text we want to locate, e.g., "the" (there should be a bunch of those in whatever Memo we chose for this exercise). Hit Enter - or click on the Match Highlight Color button with the mouse (this time, you'd click the button itself, as opposed to invoking its dropdown Color Picker). You should now see all occurrences of the text we were searching for highlighted with the color we chose - everywhere in the entire Memo (or within the

You should now see all occurrences of the text we were searching for highlighted with the color we chose – everywhere in the entire Memo (or within the selection if there was one). Not only that, but the positions of all found matches are marked with the same color on the dark-gray MatchBar on the right-hand side of the Memo. The MatchBar represents the entire length of the Memo, so some matches it might be showing may not be currently displayed if they are on another "page" of a long Memo. You can of course get to them using the regular Memo's scrollbar, but there's another way: just click on the match in the MatchBar – and the match appears right next to the point you clicked. Actually, you can click anywhere on the MatchBar, not just on matches: it can be used as an alternative to a scrollbar regardless of matches, if any.

Let's make sure there's no selection in the Memo (to search through its entirety) and let's change the Match Highlight Color to light-blue. A new MemoSearch (for some other word) – and a new set of light-blue matches appears in both the Memo and the MatchBar. You can continue with more searches and more colors, but you get the idea.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


color
Notes
Match Highlight Color
MemoSearch
Memos have full support for the common Find and Replace operations. Let's leave them for the next chapter though, and start with something less ordinary and more exciting.
MemoSearch is a way of rapidly locating text in a Memo. Here's how it works. Let's open some Memo with a few pages of text (if you don't want to "play along" just imagine you opened one). On the toolbar, let's start by selecting some Match Highlight Color by clicking the little dropdown arrow. Yellow may be a reasonable choice. Once this is done, let's move a bit to the left where we have the Match 1 combo box. This is where we shall type the text we want to locate, e.g., "the" (there should be a bunch of those in whatever Memo we chose for this exercise). Hit Enter - or click on the Match Highlight Color button with the mouse (this time, you'd click the button itself, as opposed to invoking its dropdown Color Picker). You should now see all occurrences of the text we were searching for highlighted with the color we chose - everywhere in the entire Memo (or within the

- MemoSearch allows you to perform a quick search on the entire Memo or a selection – all in "one shot".
- The matches are marked with the color of your choice and also shown on the MatchBar. You may be able to figure out what portion of the Memo you are looking for by simply observing the concentration or distribution of matches along its length.
- You can navigate the Memo by clicking wherever you want to go on the MatchBar.

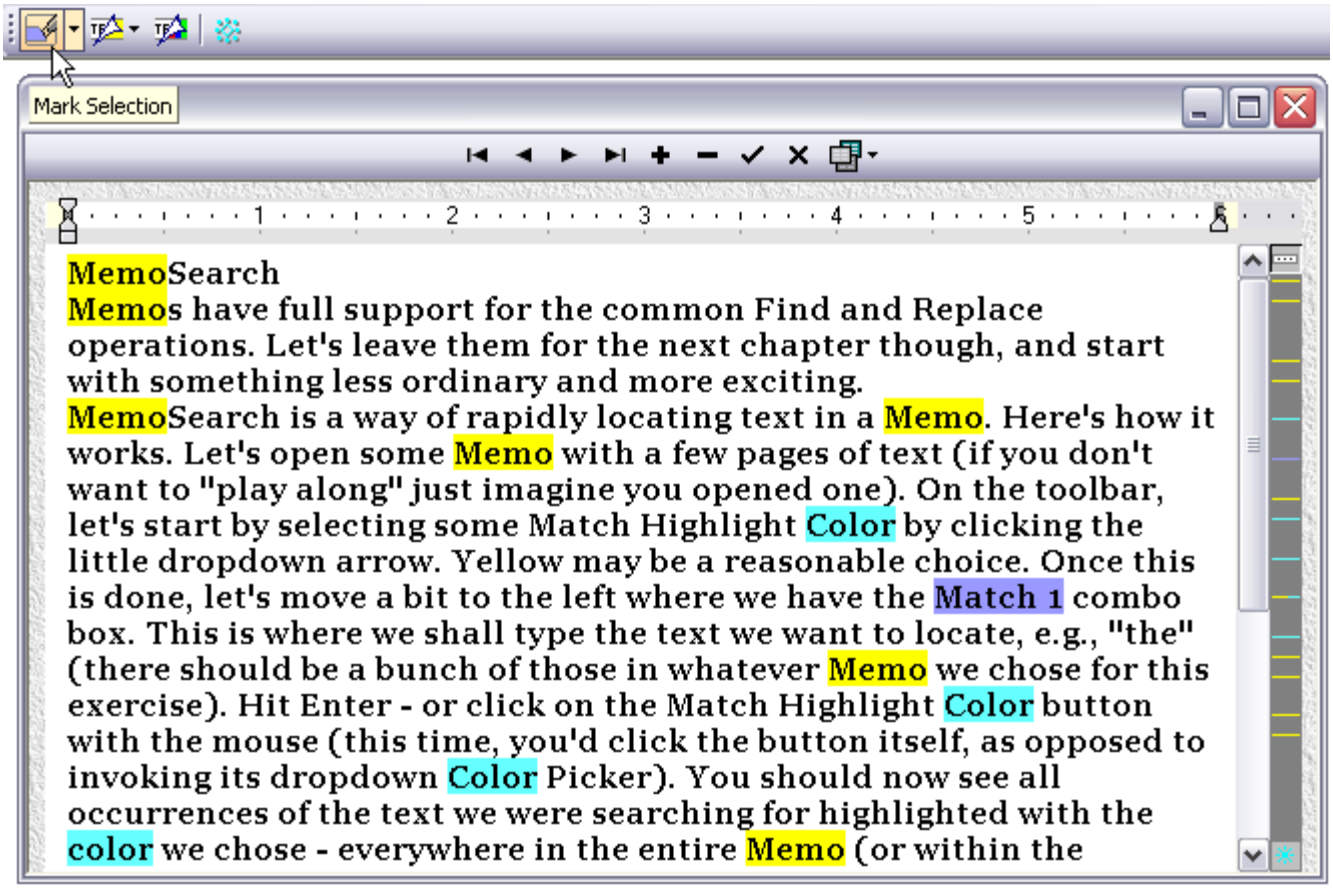
There's more though. More pairs of Match combo boxes and Match Highlight Color buttons, that is. In fact, there are 4 of them. Each pair is fully independent from the others: it can have its own favorite Highlight Color, and it remembers its own separate list of most recently used search expressions. You may never need all 4 pairs, but at least you can select your preferred Highlight Colors for, say, two of them, and use them interchangeably. This way you won't have to switch colors as often. Or you might be organized enough to want to separate your searches so that each Match combo boxes maintains some logically-organized list of relevant search expressions. Whatever your needs may be, the 4 MemoSearch pairs of controls are there for you.

That's not all. Sometimes you may want to highlight some text "by hand" without ever searching for it.

Easily done: just select the text you want and hit the Mark Selection button .
The text gets highlighted with the current Marker color of that button. Pick the color you like using the

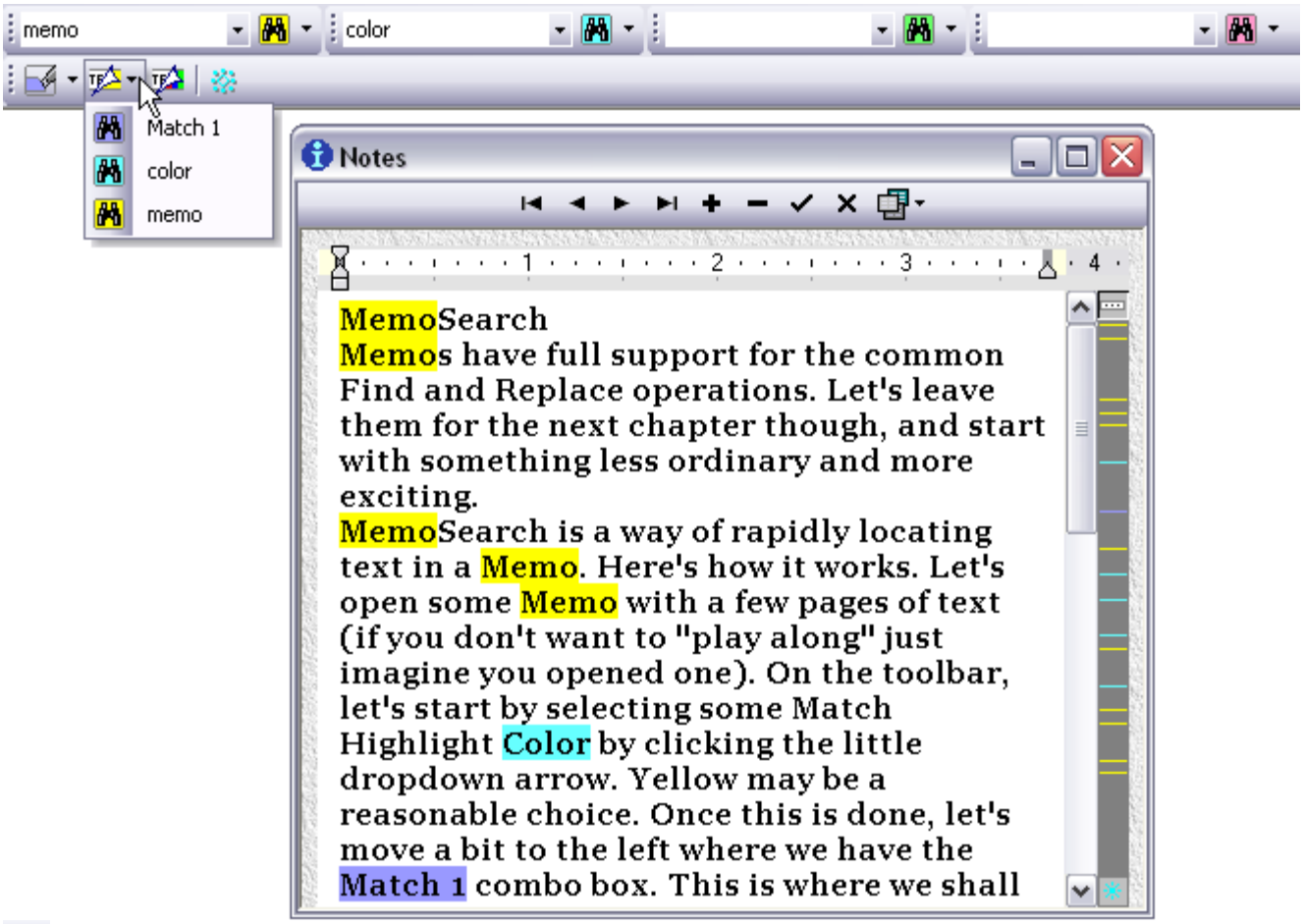
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

dropdown. Mark Selection is an InstaButton, so you can have as many of them as you like - each with its own color.


Mark Selection
MemoSearch
Memos have full support for the common Find and Replace operations. Let's leave them for the next chapter though, and start with something less ordinary and more exciting.
MemoSearch is a way of rapidly locating text in a Memo. Here's how it works. Let's open some Memo with a few pages of text (if you don't want to "play along" just imagine you opened one). On the toolbar, let's start by selecting some Match Highlight Color by clicking the little dropdown arrow. Yellow may be a reasonable choice. Once this is done, let's move a bit to the left where we have the Match 1 combo box. This is where we shall type the text we want to locate, e.g., "the" (there should be a bunch of those in whatever Memo we chose for this exercise). Hit Enter - or click on the Match Highlight Color button with the mouse (this time, you'd click the button itself, as opposed to invoking its dropdown Color Picker). You should now see all occurrences of the text we were searching for highlighted with the color we chose - everywhere in the entire Memo (or within the

Clear a Single Match lets you get rid of the match that you pick from the dropdown menu of that button. For highlighted matches produced as a result of a MemoSearch, this will clear all occurrences highlighted by it. So, if our first MemoSearch for the word "the" produced 100 matches, all of them will get cleared.

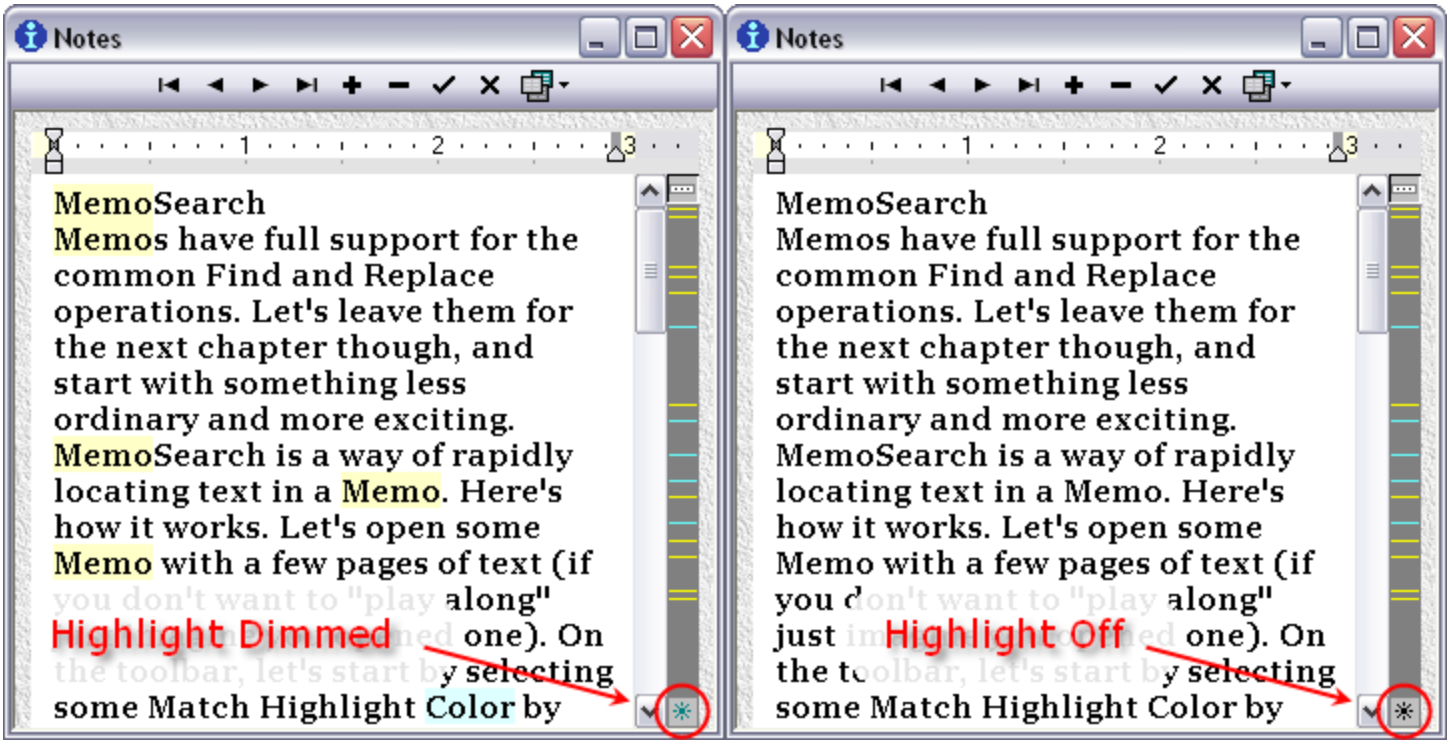
Copyright © 2002-2007 Alventis Corporation. All rights reserved.


memo
color
Match 1
color
memo
Notes
MemoSearch
Memos have full support for the common Find and Replace operations. Let's leave them for the next chapter though, and start with something less ordinary and more exciting.
MemoSearch is a way of rapidly locating text in a Memo. Here's how it works. Let's open some Memo with a few pages of text (if you don't want to "play along" just imagine you opened one). On the toolbar, let's start by selecting some Match Highlight Color by clicking the little dropdown arrow. Yellow may be a reasonable choice. Once this is done, let's move a bit to the left where we have the Match 1 combo box. This is where we shall

Clear All Matches goes even further and clears... well, all matches produced by all MemoSearches and all manual highlighting done using Mark Selection.

It's not quite over yet. Okay, so maybe you have this Memo with 10 different MemoSearches and 20 different Marked Selections. And at this point in time you don't want to see all this flashy highlighting: you just want to quietly read the Memo or maybe edit some of it. But all that highlighting has been done for a reason, and just clearing it would be wasteful. The solution is easy. Just turn off the display of the highlighting. This doesn't affect the Memo in any way, just the way the highlighting is shown on-screen. To turn highlighting On or Off, there's a little button at the bottom of the MatchBar of every Memo .

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Notes
MemoSearch
Memos have full support for the common Find and Replace operations. Let's leave them for the next chapter though, and start with something less ordinary and more exciting. MemoSearch is a way of rapidly locating text in a Memo. Here's how it works. Let's open some Memo with a few pages of text (if you don't want to "play along" just one). On the toolbar, let's start by selecting some Match Highlight Color by
Highlight Dimmed
Notes
MemoSearch
Memos have full support for the common Find and Replace operations. Let's leave them for the next chapter though, and start with something less ordinary and more exciting. MemoSearch is a way of rapidly locating text in a Memo. Here's how it works. Let's open some Memo with a few pages of text (if you don't want to "play along" just one). On the toolbar, let's start by selecting some Match Highlight Color by
Highlight Off

There's also a single Global Match Highlight Level button that will affect all Memos in all InfoViews in Alventis. Actually, both of these buttons are not On/Off buttons but rather "dimmer switches" with 3 states: Full highlighting, Dimmed, and Off. The Global Level takes precedence over any "local" Levels of individual Memos: if the Global Level is Off, for example, all Memos have their highlighting off regardless of what their current individual Level is. If the Global Level is Dimmed, Memos can't "go beyond" the Dimmed Level: they can be just Off or Dimmed. It's clearer if you just play with it a bit. Memos remember their Highlight Level setting and preserve it when closed and re-opened.
Note that since there's nothing else to see in a MatchBar, they always show all matches with Full Highlight Level.

- Are we there yet?
- No!
The act of highlighting something in a Memo is an editing operation. This means that you are actually modifying the Memo whenever you perform a MemoSearch or Mark Selection.
This also means that you can save any highlighting you have in a Memo by posting your changes. If that's what you want to do, that is. If you were just casually looking for something and don't need the corresponding highlighting, all you have to do is cancel your changes. You can of course clear existing highlighting at any time using one of the Clear buttons.

And finally, there's a little multi-user issue... In a multi-user environment, the same Memo may obviously be viewed by many users, and all of them may be searching for something, and marking it with colors that may not be to your liking, and... well, it's clear that things would very quickly get ugly or at least confusing. Which is why every user's highlighting is only visible to that user and nobody else.

This only applies to highlighting described in this chapter. All "normal" editing operations, for instance, changing text background color (which may also be seen as a kind of highlighting, but which has nothing to do with the special kind we're talking about here), all regular editing operations are

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

user-agnostic, and as such will be visible to everyone.

Congratulations! End of chapter.

2.6.14 Find and Replace

In the previous chapter we have seen how MemoSearch can help you find what you're looking for in a flash. In this chapter we will examine the more classic method of text search.

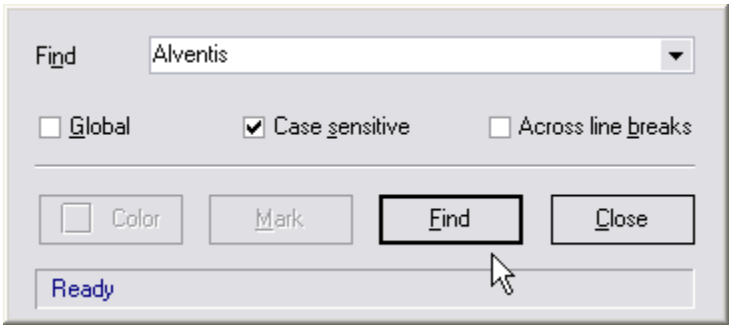
The Memo **Find and Replace** functionality is encompassed in Alventis by 3 commands and 2 dialogs:

Find in Memo – which opens the Find dialog

Replace in Memo – which opens the Replace dialog

Find Next in Memo – which doesn't have a dialog, poor thing

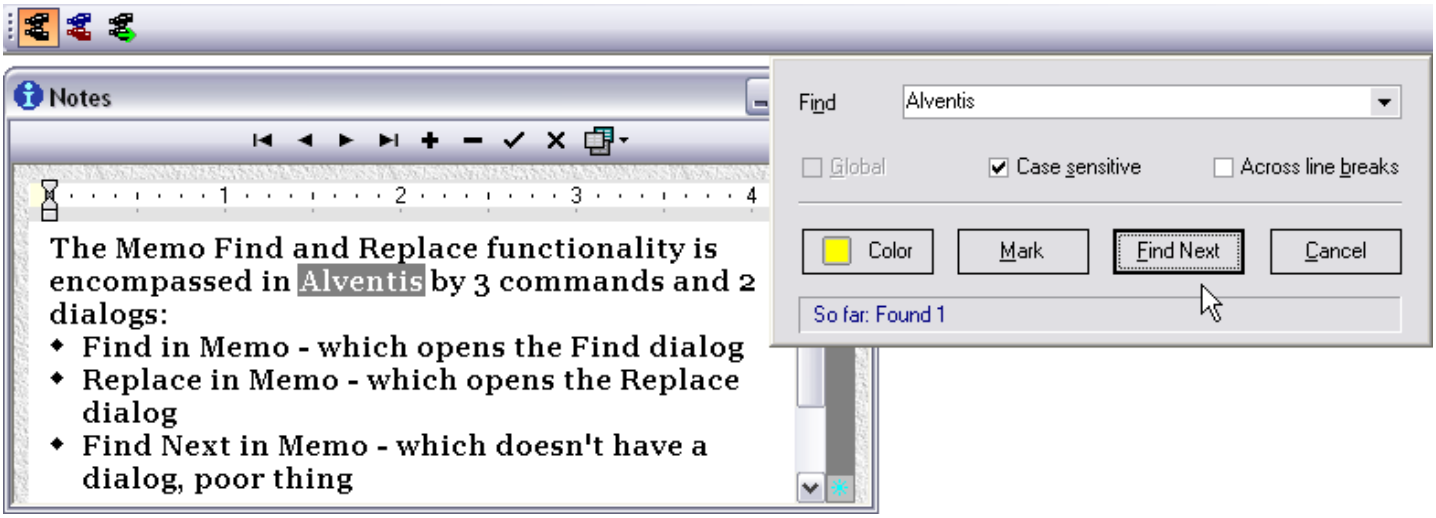
The Find dialog is depicted below.



The Find combo box is of course where you specify what text you are looking for. No regular expressions or formatting here, sorry. Just good old plain text. This is an MRU Combo Box though, so it will remember your recent searches.
The Global checkbox, when checked, will perform the search on the entire text of the Memo. Otherwise, it will start at your current cursor position in the Memo.
The Case sensitive checkbox forces the search to only find exact matches where character case is as you specified.
The Across line breaks checkbox makes the search ignore a line breaks if one is found between the words of the search expression you have specified. Could be useful if your Memo contains text with lots of manual line breaks at the end of every line – as is frequently the case with e-mails.
Close dismisses the dialog.
The Find button starts the search.

Once a match has been found in the Memo, you can stop the search using the Cancel button (the one that used to say Close) (if you've found what you're looking for). You can attempt to locate the next match by clicking the Find Next button (which used to say just Find). You can highlight the found occurrence with the Color (which you select using the so called button) by clicking the Mark button, which automatically continues the search on to the next match, if any. Highlighting has been described in the MemoSearch chapter.

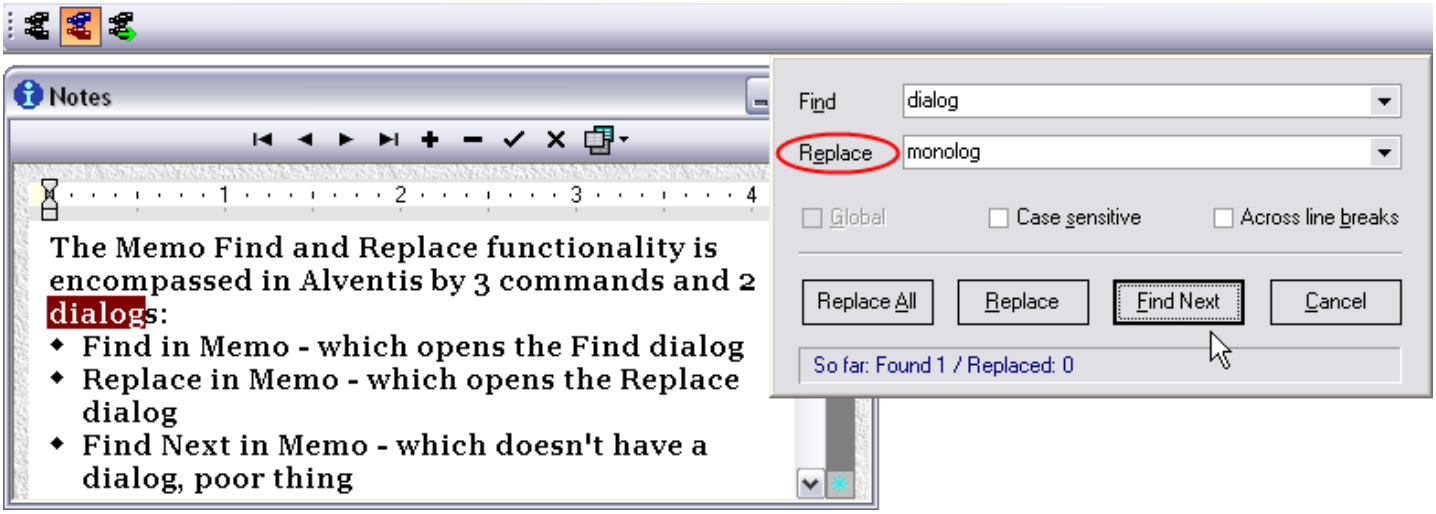
Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Notes
The Memo Find and Replace functionality is encompassed in Alventis by 3 commands and 2 dialogs:
Find in Memo - which opens the Find dialog
Replace in Memo - which opens the Replace dialog
Find Next in Memo - which doesn't have a dialog, poor thing
Find
Alventis
Global
Case sensitive
Across line breaks
Color
Mark
Find Next
Cancel
So far: Found 1

During a search, you can adjust whatever the dialog will let you, e.g., the search expression in the Find combo box. The search will continue with the updated text.
If you cancel the search in progress, you do just that: cancel the on-going search. This gives you an opportunity to perform another search without closing the dialog. If the search is Global, it will restart at the beginning of the Memo. If not, it will proceed from where you left off with the preceding search.
You can also perform a search on just the selected Memo text. You do so by first selecting the text, and then invoking the Find dialog.

The Replace dialog is quite similar to the Find dialog:

In fact, the only differences are:
- The Replace MRU Combo Box is of course where you specify the replacement text.
- The buttons related to match Marking are not appropriate here, and so have been replaced by Replace All and Replace.


Notes
The Memo Find and Replace functionality is encompassed in Alventis by 3 commands and 2 dialogs:
Find in Memo - which opens the Find dialog
Replace in Memo - which opens the Replace dialog
Find Next in Memo - which doesn't have a dialog, poor thing
Find
dialog
Replace
monolog
Global
Case sensitive
Across line breaks
Replace All
Replace
Find Next
Cancel
So far: Found 1 / Replaced: 0

The general procedure is the same as before:
Specify the Find and Replace text and set the checkboxes up as you like
Hit the Find button to initiate a search

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

If a match is found:
- Hit Replace All to replace this and all subsequent matches without any prompting
- Hit Replace to replace just this match and proceed with the search
- Hit Find Next to skip this match and try to locate another one

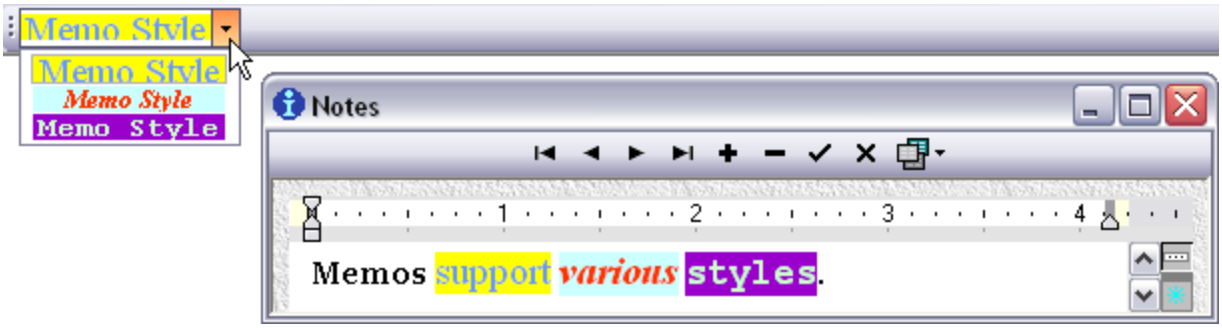
Everything we said about the Find dialog also applies here: you can perform multiple search/replace sequences without closing the dialog, you can modify whatever it will let you during a search (including, yes, both the Find and Replace text strings), and you can perform all that in a selection.

Finally, the Find Next in Memo button simply repeats the last Find operation. It simply goes from the current cursor position in the forward direction. If you haven't performed any Find/Replace operations yet since you have launched Alventis, it won't know what search to repeat, so it will simply open the Find dialog.

2.6.15 Style Picker

Alventis has rather intricate support for Styles, which is so much broader than just text styles in Memos, that we'll have to save this discussion for a chapter on Styles. What we'll do here is "scratch the surface" by mentioning just the few aspects of Styles that apply to Memos.

Alventis supports Named Styles. Their management is performed in the Style Explorer, which is fully described elsewhere. For the time being, let's just accept the fact that some styles can be named in some mysterious way.



Named styles (as well as some unnamed styles, but again, let's not get into this here) appear in the Style Picker combo box.

You can select a style and apply it to any text in a Memo just like you would apply it using the Font InstaButton or any other.

While working with your Memos, you "create" text styles using such InstaButtons as: Font, QuickStyle, Font and/or Background Colors, as well as the Bold/Italic/Underline/Strikeout buttons. Automatically adding them all to the Style Picker would very quickly make its list of styles extremely long, unmanageable, and thus not very useful. Which is why you have to explicitly add a style you want to the Style Picker by Shift-clicking it. Just place the cursor on the text stylized with the style you like, and Shift-click on the Picker.

And that's really all there is to it – as far as Memos are concerned. To give styles names and make them

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

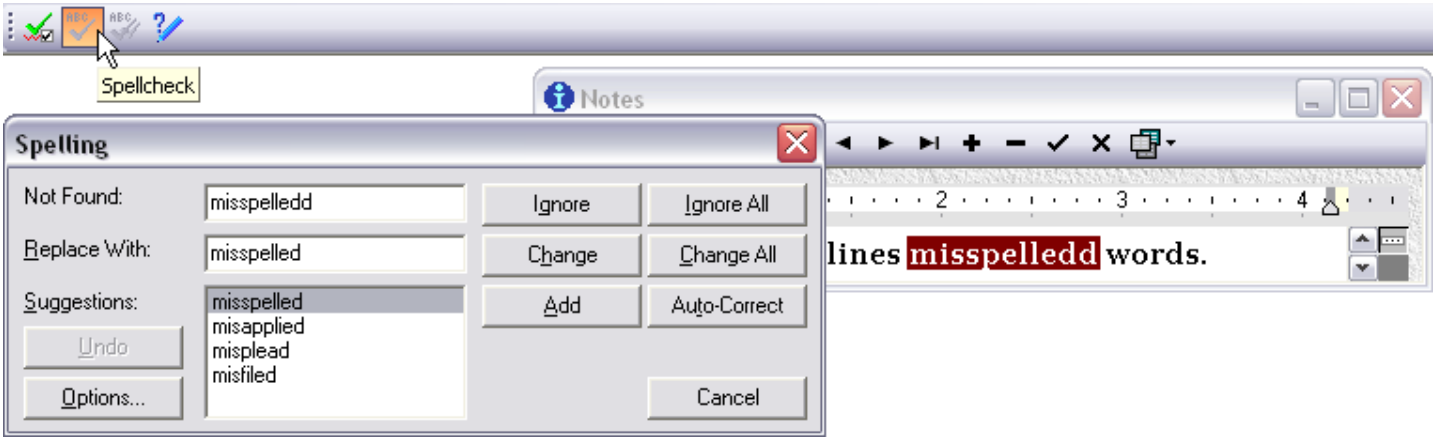
appear in the Style Picker permanently, you'll have to refer to the chapter on the Style Explorer.

2.6.16 Spellcheck and Thesaurus

Alventis supports both Spellcheck and Thesaurus. It is shipped with spelling dictionaries in American English and French, and the Roget's Thesaurus of American English. The UK English version of the Thesaurus and dictionaries in numerous other languages are available for download from the manufacturer of the component that Alventis uses for all its linguistic needs: Addictive Software (www.addictivesoftware.com/dicts-extern.htm).

The operation of both Spellcheck and Thesaurus is very similar to what can be found in most common word processors, so impatient minds can skip this chapter and learn how things work by just playing with the dialogs. We'll try to provide a brief explanation anyway.

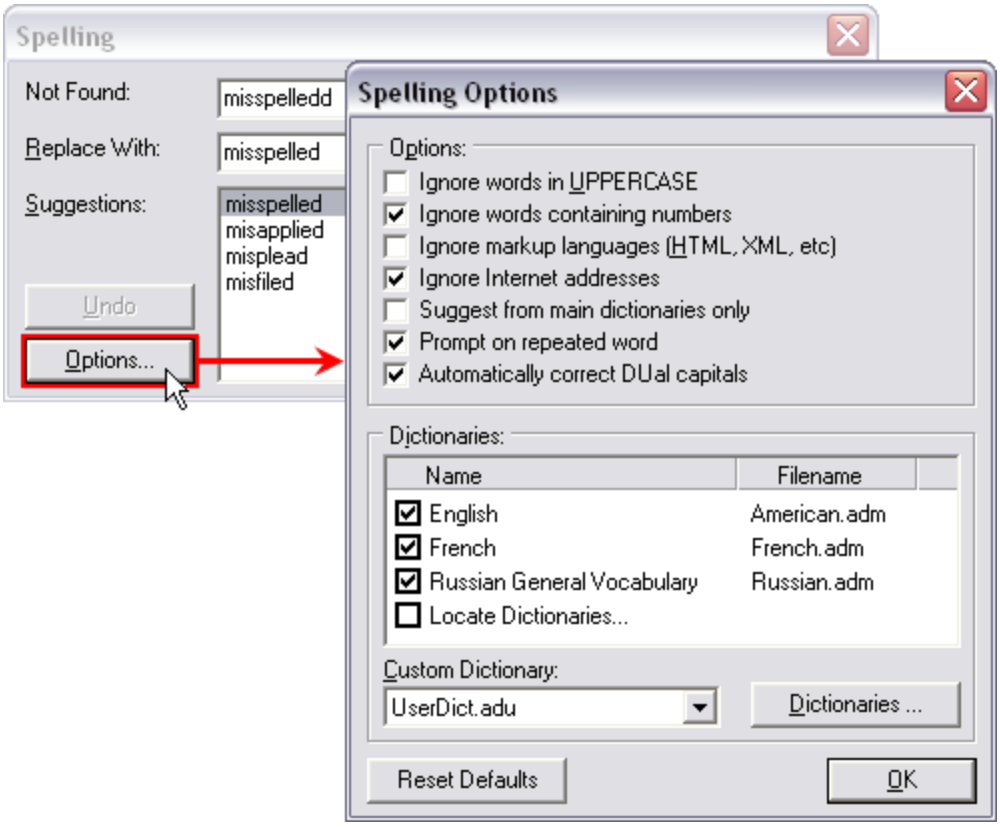
Alventis Memos support the following Spellcheck-related features:
- On-demand spellcheck via a familiar-looking dialog.
- LiveSpell which, if enabled, can underline spelling mistakes in your Memos with a wiggly red line. You may be familiar with this feature under other names, e.g., "Check spelling as you type" or similar.
- AutoCorrect which automatically replaces teh [sic] most common spelling mistakes such as the above.



Support for all this is afforded by 3 types of dictionaries:

Main Dictionaries. These are the dictionaries of some language or perhaps a thematic subset thereof (e.g., chemical terms). The English and French dictionaries fall into this category. Each dictionary comes in a separate file with an ADM extension. Each of these dictionaries can be enabled or disabled individually by checking/clearing the checkbox next to it in the Spelling Options dialog which is accessible from the Spelling dialog.

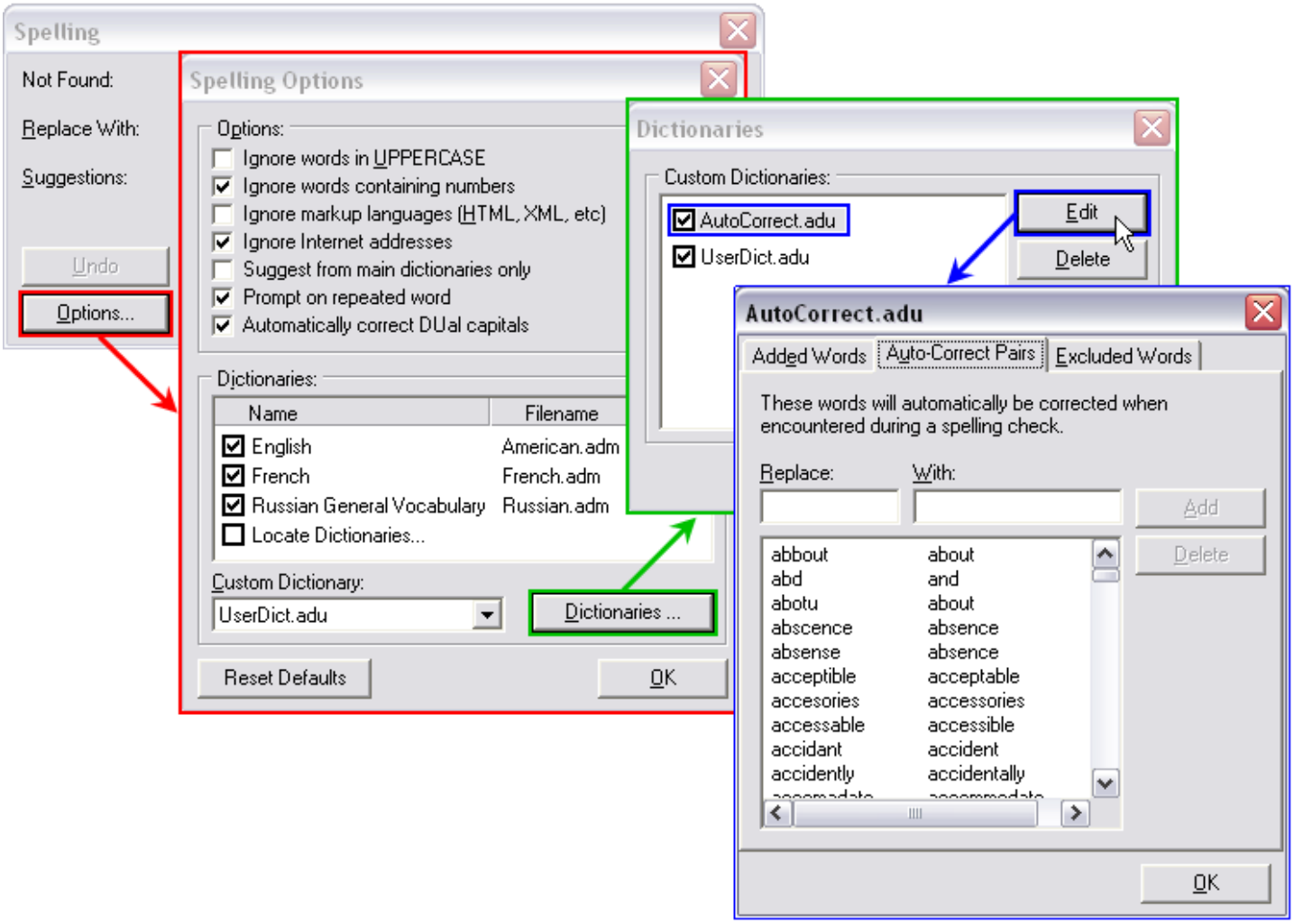
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



AutoCorrect Dictionary. There is one such dictionary shipped with Alventis. It comes in a file named AutoCorrect.adu and contains a vast array of common auto-correct pairs for the English language. More on this later.

User Dictionaries. These are the dictionaries where your amendments of the Main Dictionaries are saved. When you tell Alventis to "learn" the spelling of, say, your last name – this is where it ends up. Alventis defaults to using a single such dictionary saved in the file called UserDict.adu. Alventis will also try to use your existing user dictionaries from Microsoft Word if it finds any. They would be typically called Custom.dic. You can enable or disable the use of specific User and AutoCorrect Dictionaries. You'd do so from the Dictionaries dialog: Spelling / Spelling Options / Dictionaries. It is from here that you can view and even edit the AutoCorrect and User Dictionaries. Editing of Microsoft Word custom/user dictionaries is entirely up to you. They are basically plain-text files with one word per line, but since they are not part of Alventis, we can't take any responsibility for what you do with them, so we won't even mention them again.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



You can view or edit AutoCorrect.adu and UserDict.adu as much as you like though. They both have the same exact structure comprised of 3 sections:

Added Words. These are words that weren't found in any of the Main dictionaries, were therefore flagged as misspelled, to which you at some point responded "Add" in the Spelling dialog. This action adds words to this section of UserDict.adu. Which is why this section is likely to be empty for AutoCorrect.adu. This section can be thought of as an extension of the Main dictionaries.

Auto-Correct Pairs. This section contains pairs of words: the misspelled and the properly spelled version. For example, teh / the or adn / and. This section will be quite lengthy in AutoCorrect.adu since that's precisely its raison d'être. You can add your own favorite pairs there, or better yet, in UserDict.adu which is likely to have no pairs there yet, so it'll be easier for you to manage your AutoCorrect setup. Pairs are automatically added there whenever you click the Auto-Correct button in the Spelling dialog. This feature is of course not limited to correcting common misspellings. You can, for instance, tell it to replace occurrences of xxa by your signature or address or any other text you have to type frequently. As soon as you type xxa and hit either Space or Enter, it will be replaced by the text of your choice.

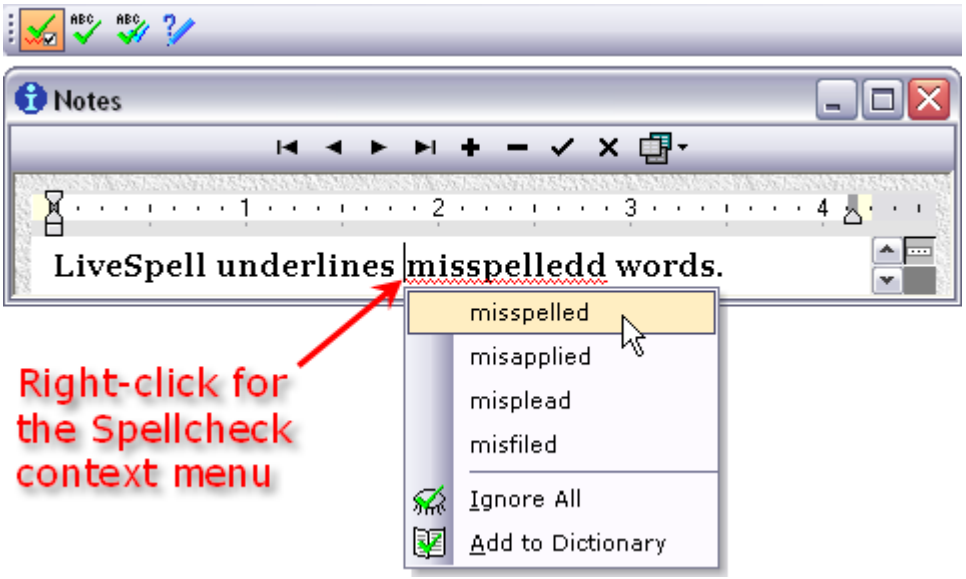
Excluded Words. This lists words you want always considered misspelled. Foul words, perhaps. Maybe you can come up with a more creative use for this.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The words you choose to simply Ignore or Change in the Spelling dialog are not remembered since both of the above operations are "one shot deals". Words you elect to Ignore All or Change All must be remembered somewhere to not prompt you again if they are encountered later. They are temporarily stored (just until you close Alventis) in a pseudo-dictionary called "(Ignore All / Change All)" which appears in the list of other User dictionaries whenever there's something in it. You can edit it in much the same way you'd edit the other "real" ones.

LiveSpell. We already told you what it is at the very beginning of this chapter, and there's really very little else to say. Whenever the Spellchecker considers a word misspelled, it will "flag" it with a wiggly red underlining – just as it is done in quite a few modern word processors.

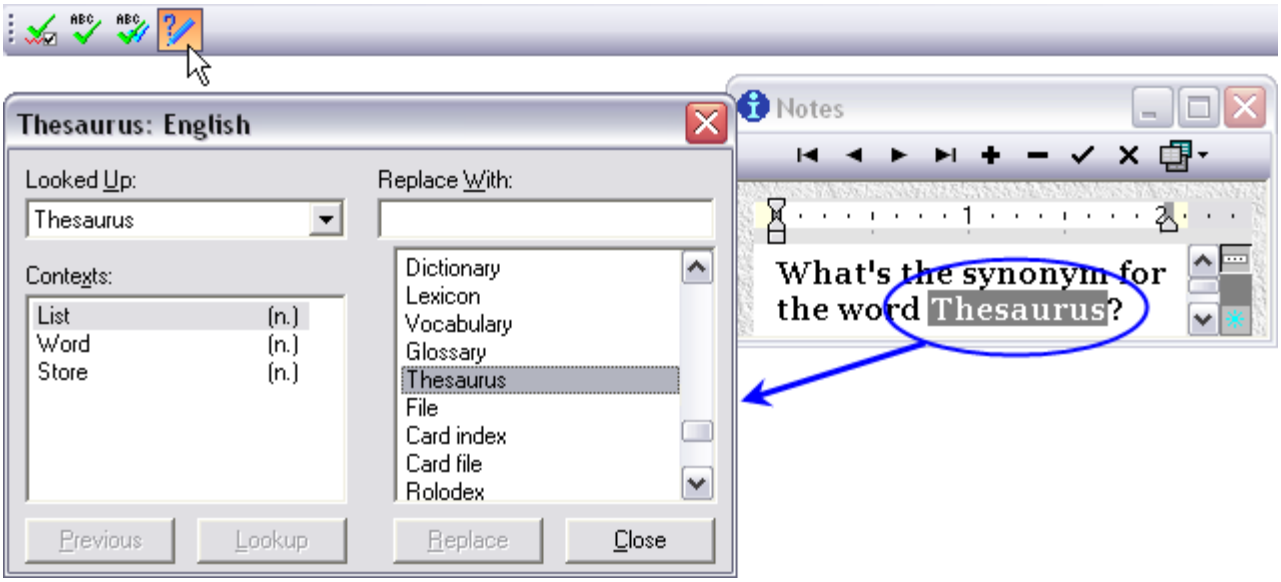
You can right-click the misspelling and "make things right" using the context menu that appears. At the top it will list the spelling suggestions, if any. Picking one will correct the misspelling. Beneath these you will find two buttons: Ignore All and Add to Dictionary. These are equivalent to Ignore All and Add buttons in the Spelling dialog. Note that as is commonly the case, you can invoke the context menu by placing the cursor on (or just to the right of) the misspelled word and hitting the Application key on your keyboard. That's the one usually sitting right next to the right Ctrl key.



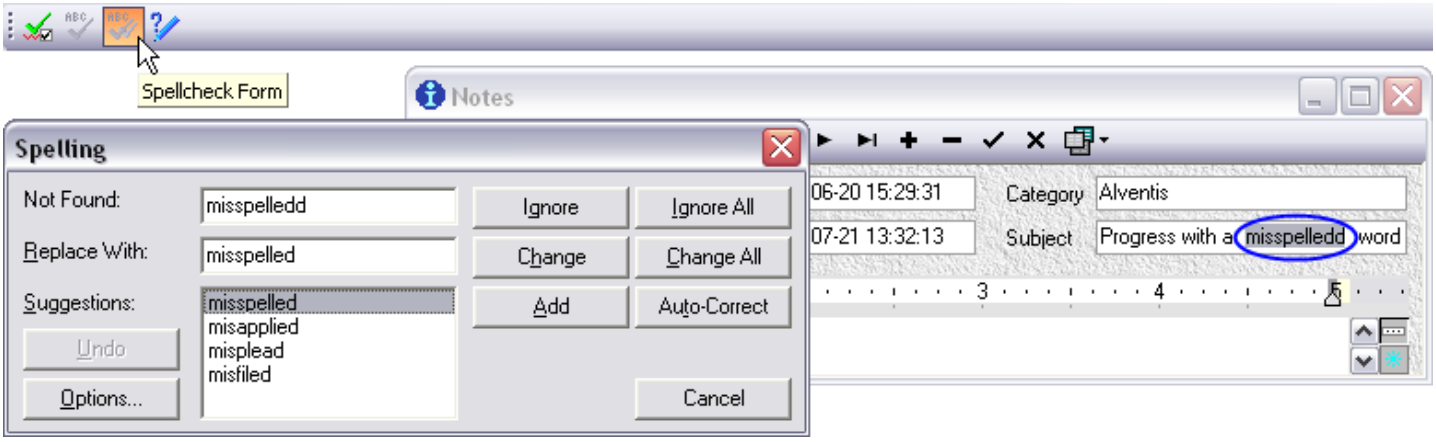
If you ever want to turn LiveSpell off, doing so is very easy: just toggle it by clicking the LiveSpell Toggle button. This setting is global, i.e., it will affect all Memos within Alventis.

Thesaurus. Select a word or simply place the cursor on it and invoke the Thesaurus to get a list of synonyms for that word. The interface is very simple and should be familiar to you from other word processors, so we will forego further detail.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Spellcheck Form. Saved for last is a little surprise. In Alventis, both Spellcheck and Thesaurus work in not just Memos, but other controls as well: Text Boxes, for example. And since you may have significantly more than one Text Box on an InfoView form, you can Spellcheck them all one after another with a single command: Spellcheck Form.



2.6.17 Drag-and-Drop

Memos support a variety of **Drag-and-Drop** operations. Most of them work exactly as you'd expect from a modern word processor, so we won't dwell upon them. We will briefly point out what we believe may be unfamiliar to you.

You can drag-and-drop text between Alventis Memos and other applications. E.g., WordPad to Memo and vice versa.
We already mentioned that you can drag-and-drop selections of text and/or hyperlinks from Microsoft Internet Explorer to Memos.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

You can also drag-and-drop files into a Memo. Files whose format is directly recognized by Alventis get their contents inserted at the point where you drop them. Such files are: Text (TXT), Rich Text Format documents (RTF), all supported Picture files: Bitmaps (BMP), JPEGs (JPG/JPEG), GIFs (GIF), PNGs (PNG), Icons (ICO), Enhanced and regular Metafiles (EMF/WMF). Attempts to drag-and-drop an unsupported file will be ignored by the Memo.

As was described in the chapter on Hyperlinks, you can insert a link to any file (regardless of its format/extension) into a Memo using Alt-drag-and-drop, i.e., hold down the Alt key while dropping.

The same chapter on Hyperlinks also explained how you can create links to records by dragging them from the Search Results into a Memo.

2.6.18 Other Operations

You may be familiar with most items in this topic from your experience with other word processors, so we'll just list them with brief descriptions.

Format Painter. Place the cursor on some text whose format you would like to replicate to some other text. Click the Format Painter – it now "remembers" the chosen format and becomes active (its button is depressed). Select the text you would like to apply your chosen format to: the just-selected text becomes formatted, and the Format Painter gets automatically deactivated.

If you would like the Format Painter to not get deactivated and remain active so that you could format more than one selection, use double-click in place of a single click that you saw in the above instructions to activate the Format Painter.

To manually deactivate the active Format Painter simply click it with the mouse or hit Escape.

Indent/Outdent. These buttons move the left indent of the selected paragraph(s) to the next/previous Tab stop respectively. When working with a Bulleted/Numbered List though, the same commands promote/demote the selected lines to the next/previous B&N Level.

Paragraph Alignment. This is a group of radio buttons that set the alignment of the selected paragraph(s) to Left/Center/Right/Justified respectively.

Undo/Redo. Memos support Undo for most operations. See the keyboard shortcuts below.

Select All. Selects all text in a Memo. Also works in UniGrids to select all Search Results records (you

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

must be in Search Results already).

Clear Styles. Clicking this button simply returns the selected text formatting to default "no particular" formatting.

B/I/U/S. Bold/Italic/Underline/Strikeout. Need we say more?

Font/Size. These combo boxes allow you to set the Font Name and Size directly.

Insert Date/Insert Time/Insert Date and Time. These buttons will insert what their names suggest in either a Memo or a Text Box in an InfoView form.

Insert Page Break. Inserts a "hard" page break in the Memo. This of course only affects printing.

Memos also have some useful **keyboard shortcuts** that perform some equally useful operations:

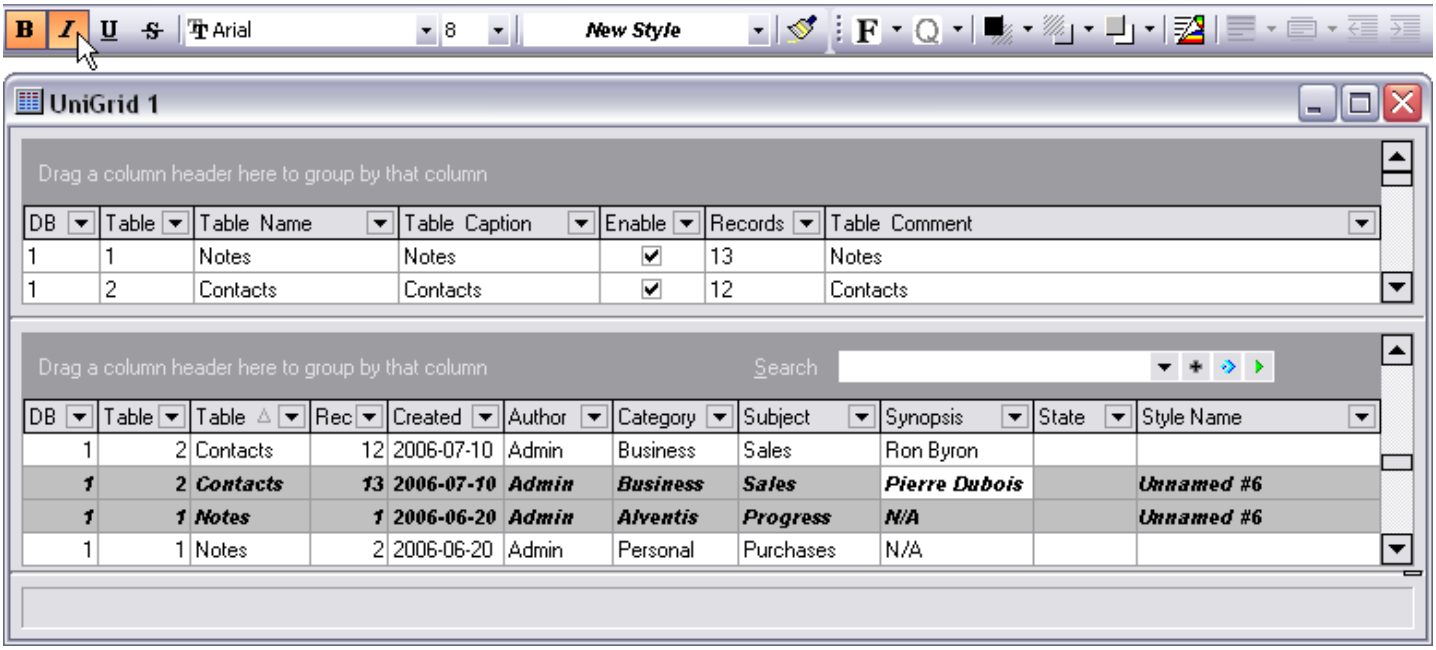
Shift-Enter	Insert soft line breaks. Such breaks start a new line but (in contrast to ordinary paragraph breaks you obtain when you hit Enter) not a new paragraph, so paragraph formatting applies to the whole block of text even though it has such line breaks. This can be particularly useful if using first line indents.
Ctrl-Z and Alt-Backspace	Undo.
Shift-Ctrl-Z and Shift-Alt-Backspace	Redo.
Ctrl-Backspace	Delete the word to the left.
Ctrl-Delete	Delete the word to the right.
Ctrl-Shift-Delete	Delete from carret until the end of the paragraph.
Ctrl-Up / Ctrl-Down	Scroll the document up or down respectively.

2.7 Record Styles

Data records in Alventis can have Styles applied to them. The Styles we are referring to are purely visual and consist of the most common Font attributes: Font Name, Charset, Size, Bold/Italic/Underline/Strikeout style, Text and Background Colors.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

While it is true that such visual Styles do not carry any well-defined "information" attached to them, they can still be effectively used as a kind of pseudo-data flags or qualifiers attached to record. For instance, you may set the background of all work-related records from all tables to some adequately repulsive color, and paint all entertainment-related ones in some other way. This allows you to quickly identify records by the visual queues their Style conveys to you. You can also sort and group them based on their Style, but let's not jump ahead of ourselves.



Stylizing records is done in much the same way you'd do it if they were simple lines of text in a Memo: select the records you want in the Search Results grid and use all the text-style tools available: Font, QuickStyle, Colors, Clear Text Formatting Style. Even the Format Painter. The latter can "copy" styles not only within Memos or between records, but between Memos and records too.

Styles that you assign to records are automatically added to the Style Picker (unlike the Memo whose styles you must add manually).

Every new style you apply to a record is given a default name, such as "Unnamed #123". This name appears in the Style Name column of the Search Results grid. If you apply to a record a style that already exists, the existing style is re-used. For example, if you make some record that didn't have any style Bold, its style may become "Unnamed #5". If you make some other record Bold, the second record will simply re-use the style of the first one, and it too will end up with the "Unnamed #5" style.

You can rename styles using the Style Explorer, which is discussed in its own chapter.

As we have mentioned earlier, once you have stylized some records, you can sort them and group them by their style using the Style Name column of the Search Results grid. You can also filter by style of course. All regular grid operations are supported. This gives you the ability to perform the above operations across the table boundaries. If you are, for example, marking all health-related records in a particular style, such records will be grouped together in the Search Results no matter which table they are coming from and what they contain: a record of your dentist in the Contacts table or a record of a healthy recipe in the Recipes table.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

You can also stylize an entire table with a Default Table Style. Just select the table in the Tables Grid and use all the usual tools. Same goes for entire Databases as described at the very end of the Database Explorer chapter.

Record Styles are recorded on a per-user basis. That is, whatever styles you paint your records with, they are only visible to you and nobody else (if there are multiple users accessing the record/table/database in question). This is similar to how each user only sees his own Memo highlighting, remember? The exception to this rule is a feature in its own right, which is discussed below.

The following discussion is mainly applicable to Alventis used in a multi-user environment, but even a single user can draw some benefit from this advanced feature, so you may want to read on.

Administrative Styles. First, let's remember that users in Alventis exist in a hierarchy, a kind of "pyramid of power". Suffice it to say that we may be looking at such users as, for instance, an "Employee" and his "Manager". We'll use these two as an example for this explanation. Normally, the Manager can mark records with some styles, and the Employee can mark them in some other way, and each of them only sees his/her styles. In addition to this "user-style independence", it is possible for the Manager to apply a style in such a way that it will be visible to all of his subordinates. Such styles are called Administrative Styles. A user will see his own or his superior's Administrative Styles marked with the symbol in the Style Name column.

The basic rule is rather simple, actually. Ordinary, personal styles are only visible to their creator (the user who applied this style to the record, that is). Administrative styles are visible to their creator and all his subordinates in the user hierarchy.

Arbitration. The Employee can apply his preferred style "over" his Manager's Administrative style (we are of course talking about styles applied to the same record(s) here). The Manager's style is still intact, and still visible to his other subordinates, but not to our clever Employee. Imagine the following scenario. The Manager wants to bring some important piece of news to the attention of his department, so he marks the corresponding record in flashy red. The Employee has seen it, diligently read it, and doesn't really want or need to see this record in red any more, so he can "paint over it" using whatever style he prefers. Or not using any particular style. He can simply clear the Manager's style using the Clear Text Formatting Style button . This does not affect the Manager's style in any way, but from the Employee's perspective, this record no longer has a style. What happens in reality, is that the Manager's style is painted over using a neutral style that makes the record look as if it had no style at all.

But what if the next day there's an important update to that same piece of news in that same record, and the Manager would like to once again inform his team about it? He'll give the record an updated Administrative style, perhaps leaving the red background and making the text Bold, but what about our Employee who has "painted over" this record's style? He won't be left out either. Since the Administrative style is newer than whatever style the Employee has applied, it's the Administrative style that "takes over".

So, the Arbitration rule is also quite simple: if an Administrative style and a Personal style are applied to the same record, the more recent style application "wins".
This stylizing "war" can go back-and-fourth between the Manager and the Employee for as long as they want. Or until retirement.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

So now you know what Administrative styles are, but we still haven't told you how to apply them. This is easy.

The Admin-Style Mode toggle button , when depressed, places you in a mode where whatever styles you apply (using all the usual techniques) become Administrative. You must therefore be diligent and turn this mode Off as soon as you're done. Otherwise your subordinates will end up seeing all your styles – unless that's what you want of course.

Since Administrative actions on your part affect other users, you may or may not have the Right to go into the Admin-Style Mode. The right to do so is part of Alventis' Multi-User Security system, and as such is only relevant to the Enterprise version.

There's one last thing to be aware of. The Manager himself sees both his Personal and his Administrative styles, and can arbitrate between them. For example, he can Administratively mark that news record red for his Employee, and then make that record green for himself with his Personal style (after turning Off the Admin-Style Mode of course).

2.8 Hidden/Sticky Records

The idea is quite simple, so we'll just say it:

- **Hidden records** are those you don't want to see in the Search Results (even if they ought to be there).
- **Sticky records** are those you always want to see in the Search Results regardless of what the current search may be.

Sounds easy? It mostly is.

You mark records Hidden or Sticky using the Make Sticky and Make Hidden buttons respectively. These are toggle buttons, so you can unmark a Sticky record by un-clicking the Sticky button. You can also clear either marking using the Clear Sticky/Hidden button.

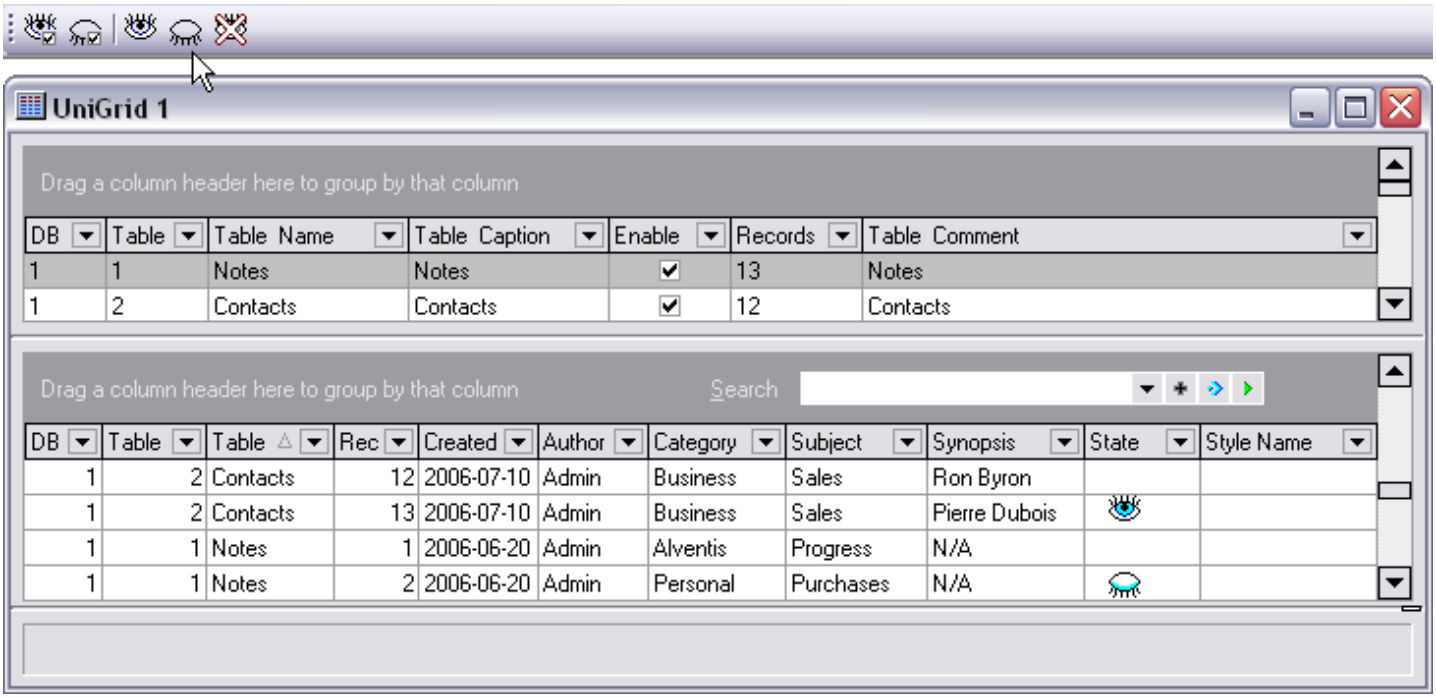
You can also enable or disable the effect of each type of marking:

Use Sticky is a toggle button that, if depressed, will make Sticky records "do their thing" and remain in the Search Results grid all the time. If this button is in the Off state, Sticky records are still marked as such, but they behave like any other record.

Use Hidden is similarly responsible for the behavior of the Hidden records.

If you ever wanted to clear the Hidden flag from a record, you'd obviously need to temporarily turn Use Hidden Off, find the record in the Search Results, select it, and clear the Hidden flag.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


UniGrid 1
Drag a column header here to group by that column
DB | Table | Table Name | Table Caption | Enable | Records | Table Comment
1 | 1 | Notes | Notes | 13 | Notes
1 | 2 | Contacts | Contacts | 12 | Contacts
Drag a column header here to group by that column
Search
DB | Table | Table | Rec | Created | Author | Category | Subject | Synopsis | State | Style Name
1 | 2 | Contacts | 12 | 2006-07-10 | Admin | Business | Sales | Ron Byron
1 | 2 | Contacts | 13 | 2006-07-10 | Admin | Business | Sales | Pierre Dubois
1 | 1 | Notes | 1 | 2006-06-20 | Admin | Alventis | Progress | N/A
1 | 1 | Notes | 2 | 2006-06-20 | Admin | Personal | Purchases | N/A

And this would be the end of that Hidden and Sticky subject, if it weren't for one other aspect of it.

Administrative Hidden/Sticky. If you've been reading this manual consecutively, you already know everything about it. If not, we'll just say here that the Hidden/Sticky attributes of a record work very much the same as Record Styles, including the fact that Hidden/Sticky flags can be applied Administratively to be seen as such by a user's subordinates.

Arbitration also works the same way. So we won't repeat our tedious explanation here and instead refer you to the portion of the chapter on Record Styles that deals with Administrative Styles and Arbitration between them.

UniGrids indicate the Hidden/Sticky states of records using the following symbols in the State column:

- Hidden by User (that'd be you)
- Sticky by User
- Hidden by Administrator
- Sticky by Administrator
- Hidden by User, with some arbitration between User's and Administrator's state going on
- Sticky with arbitration
- Cleared by User (i.e., the User cleared whatever state has been set by the Administrator)

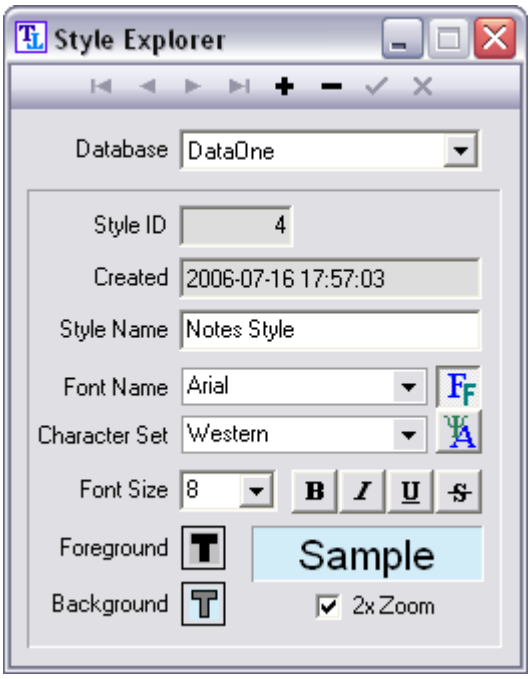
2.9 Style Explorer

You may have learned a thing or two about Styles in Alventis from the previous chapters on Record Styles, the Style Picker, and Memo formatting. Here, we will discover how you can manipulate all these

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Styles directly.

The **Style Explorer** form is depicted below.



We said "form" and not "dialog" on purpose. The Style Explorer can remain open while you work in some other form in Alventis. You can close it when you don't need it any more.

The Style Explorer is quite straightforward. At the very top you find the DataNav bar that should be familiar to you from the InfoView chapter. It lets you navigate between Styles of the Database (that you'll select in a moment). Styles are treated here just like data records, so if you know how to use an InfoView, there should be no surprises for you here.
Right underneath it is the Database combo box. Styles in Alventis are considered part of the Database whose record(s) they are applied to. This combo box allows you to choose which Database's styles you want to explore.

The remaining controls essentially comprise the individual Style itself.
Style ID is just a unique style order number automatically assigned at the time of its creation (just like a data record's ID).
Created is the style's creation date.
Style Name gives you an opportunity to name the style any way you want for identification purposes. This name is your "personal" name for the style, i.e., it is not visible to other users, if any. The Style Name has more importance than we just led you to believe though. It tells Alventis that this style has some "special meaning" for you and should therefore be preserved for posterity – even if it is not currently applied to any data records. Alventis only preserves styles that are either currently applied to at least one record or that you have explicitly named. This is where you do so. It is only these preserved styles that will appear in the Style Picker when you launch Alventis.

Note that while unnamed styles are displayed in the Search Results grid as "Unnamed #123" or similar (123 being the Style ID in case you haven't guessed), these "fake" names are not listed in the Style Explorer.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Note also that the fact that a style may be "used" somewhere in Memo text does not play any role and won't make such styles preserved. The text in the Memo remains stylized of course, but the style will not appear in the Style Explorer or the Style Picker.

The rest of the controls look and work almost exactly like their counterparts in the Font dialog box, so please refer to that chapter if you have some doubts. You can use them to modify an existing style or create a brand new one.

2.10 Synopsis Workbench

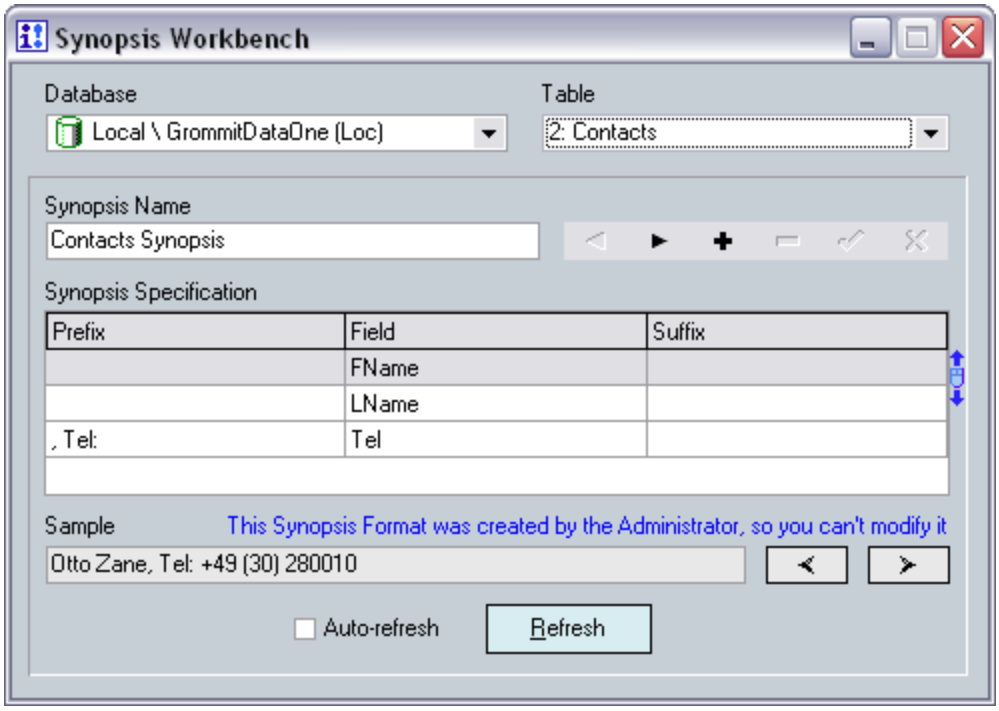
We have briefly discussed the Synopsis column when we were talking about the Search Results grid. We will now discover where all this Synopsis text comes from.

The **Synopsis Workbench** form is depicted below.

The Synopsis Specification of a table is basically a sequence of fields of that table interspersed with whatever text might be suitable. Let's clarify this with some examples. Imagine we are talking about the Contacts table that stores names, addresses, etc. We'll see all that information in all its glory in the appropriate InfoView of that table. The UniGrid's Search Results are unfortunately much more terse, being limited to directly displaying only the values of the always-present System fields. So, if Category and Subject are not sufficient for properly identifying the record in the Search Results, we have to resort to the Synopsis.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

For a Contact record, for example, we may want to see at least some information about the person the record describes. Something like "John Smith" would be nice. Synopsis allows us to do just that (and more). Note that "John" and "Smith" were likely coming from different fields of the Contacts table (First Name and Last Name respectively).



The truth of the matter is, we can put whatever string/text or numeric fields we want here, so we can make the above Synopsis even more useful by including the person's telephone number like so: "John Smith: 321-555-3210". Whatever fits in the Synopsis column.

Let's see now how we would achieve the above.

First, we need to select the Database and the Table whose Synopsis we want to manipulate. The two combo boxes at the top do just that.

Most of the remaining controls correspond to a single Synopsis. You can think of it as a Synopsis Record. And you can have more than one. That's right, a table can have as many Synopses as you want. You can navigate between them using the trimmed-down version of the DataNav bar.

Let's go ahead and create a new Synopsis by clicking the Plus button on the DataNav bar. A new blank Synopsis is created for us and is assigned a default Synopsis Name. Feel free to change it to something more memorable or leave it as is.

Right beneath it we find the Synopsis Specification grid. This is the "heart" of this form as this is where we arrange the fields we want in the order we like. Focus the grid by clicking on it (you may not see any change if the grid is empty). As with most grids in Alventis, create a new Spec record by hitting Insert on the keyboard. A new blank record appears in the grid. Let's start with the Field column and set it to FName (First Name). It's a combo box column, and we can only pick fields that are appropriate for a Synopsis. System fields are not listed because they already appear in the Search Results. Memo and Graphic fields can't be shown as part of a string of plain text, so they don't appear either. As soon as you pick a field, the Sample Text Box is updated to reflect your changes, so by now it may be displaying the First Name of the person from whatever happens to be your first record of the Contacts table. Hit Insert again to add a second Spec record and set its Field to Last Name. The Sample becomes something like "JohnDoe". That's not quite right, but let's add the 3rd field we want in there to get it

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

over with: the Telephone number, so we'll add the Tel field in a similar fashion. We now end up with "JohnDoe123-555-4444" or something like that – depending on which data record of the table provides us with the data for the Sample. If the first data record doesn't suit you (maybe its relevant fields are empty), you can try to navigate to a better one using the Next/Prior buttons right next to the Sample. Let's fix our Synopsis though.

All we need to add is some punctuation to make the Synopsis more readable. We can put a single space character either in the first field's Suffix or the second field's Prefix, and the ": " pair in either of the two between the Last Name and Tel. Both Prefix and Suffix do basically the same thing: their text is inserted according to their position in the Synopsis Spec, so you can use whichever you prefer. Using both for the same field may make sense if you want to enclose its value in parenthesis or similar, e.g., "Smith (John)". You can even use either or both and leave the Field column blank if that's what you want.

Reordering Spec records is easy too: just drag them up/down with the mouse. Note that this automatically saves your changes.

If the Auto-refresh checkbox was checked, as soon as you post your changes to a Synopsis that is currently used (i.e., displayed) in an open UniGrid's Search Results, they will be automatically applied to wherever they are visible. Otherwise, you can click the Refresh button and it will refresh the Search Results. If you don't refresh them from here, they will get refreshed the next time you perform a search.

Alternative Synopses.
We've mentioned earlier that a table can have more than one Synopsis, but what do you do with them? If a table has more than one, you can simply select which one you want to use. You do so directly in the Search Results, but there's a trick to it. You need to Alt-click the Synopsis of the table you want to change it for. This will "let you into" the corresponding cell. You can now pick the Synopsis you want from the combo box.

You can also click the little dotted button at the right to open the Synopsis Workbench and display the currently selected Synopsis.

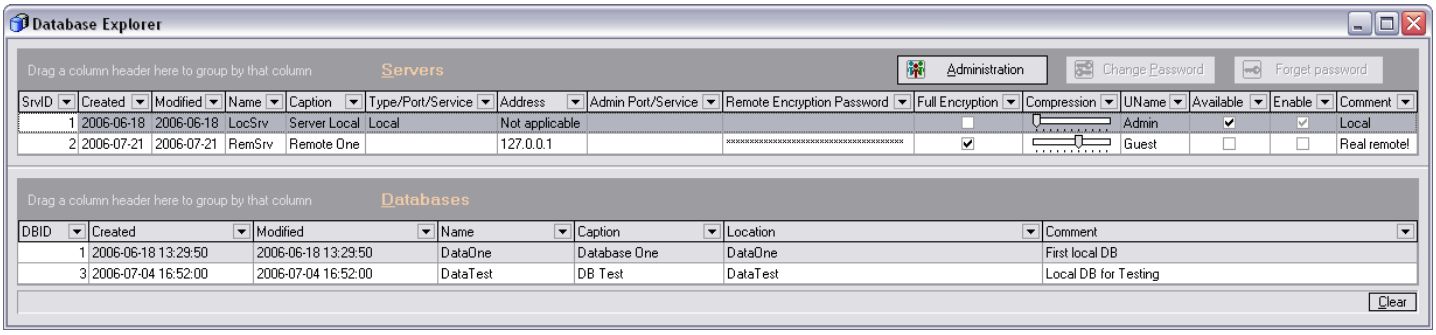
Your choice of which Synopsis to use is your personal business – in the very specific sense that in a multi-user setting it does not affect the other users. The Synopses themselves are shared between all users though, so be careful if editing existing ones.

2.11 Database Explorer

We discuss the general concepts relating to Servers and Databases in a chapter dedicated to them, so feel free to refer to it if necessary.

Database Explorer is a simple-looking form that lists all Servers and Databases you can access.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Database Explorer
Drag a column header here to group by that column
Servers
Administration
Change Password
Forget password
SrvID | Created | Modified | Name | Caption | Type/Port/Service | Address | Admin Port/Service | Remote Encryption Password | Full Encryption | Compression | UName | Available | Enable | Comment
1 | 2006-06-18 | 2006-06-18 | LocSrv | Server Local | Local | Not applicable | | | | | Admin | | | Local
2 | 2006-07-21 | 2006-07-21 | RemSrv | Remote One | | 127.0.0.1 | | ********** | | | Guest | | | Real remote!
Drag a column header here to group by that column
Databases
DBID | Created | Modified | Name | Caption | Location | Comment
1 | 2006-06-18 13:29:50 | 2006-06-18 13:29:50 | DataOne | Database One | DataOne | First local DB
3 | 2006-07-04 16:52:00 | 2006-07-04 16:52:00 | DataTest | DB Test | DataTest | Local DB for Testing
Clear

The form is split in two with a Splitter that allows you to adjust its layout to some extent.
In the upper portion of the form you find the Servers grid, and in the lower portion – the Databases grid. At the very bottom there's a Status bar with a little Clear button to clear, if you so desire, messages that may appear there once in a while.

Servers Grid.

The Servers grid lists the servers known to Alventis. Let's clarify something right away. Alventis can function in two modes:

- Local mode is when Alventis is accessing data directly. The data may physically reside on your computer or on another computer on the network, but in either case Alventis is accessing it as one would access files in Windows.
- Client-Server (C/S) mode is when Alventis is making requests to a Server (which is a standalone application) without directly "touching" the data files. More details on all of this are in the Servers and Databases chapter.

What's important to realize is that Alventis can use both of the above modes concurrently, i.e., access some Databases locally, and communicate with remote Servers to access other Databases. Aside from the admittedly most important question "where is the data?", the rest is very similar no matter which access mode is in use. Which is why we will be referring to all data you can access locally as a "Local Server" – even though there is no server per se in this case.

Peer-to-Peer Shared Access. This is not a separate mode, but is nonetheless worth explaining in case you need it. As we have mentioned, Alventis can use Local mode to access data residing on another computer on the network. More than one user can access such data concurrently. Such Shared Access can provide multi-user operation for small groups of users that do not require the benefits of real Client-Server operation. Note that C/S mode is more robust since all data requests are "funneled" through a single Server application, presumably running on a secure, undisturbed, robust computer with adequate backup power protection, etc. And only C/S mode is truly multi-user in the sense of maintaining multiple user accounts, access rights and privileges, and so on.

Back to the Servers Grid now. Each line in the grid corresponds to a Server. Alventis will normally list at least one Local Server here, and in addition to that, you may add any number of Servers known to you. You would add a server by simply inserting a new record in the grid. Let's quickly go over its columns.

SrvID is just the order number of the Server record, it has no special meaning and no relation to the Server itself, so you can ignore this number for the most part.

Created and Modified are likewise, dates this record was created/modified (not the Server itself).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Name, Caption, and Comment are there mostly for your convenience: you can name the Server as you see fit, and leave yourself a comment if you wish: none of this is related to the Server itself, so enter whatever you want here.

Address. At long last we get to something that matters! If this is a record for a real remote Server (accessed in Client-Server mode), this is where you enter its address. C/S mode uses the TCP/IP protocol – same as used on the Internet, and the address of the Server must be specified the same way you'd specify the name of a Web server. It can be the name of the machine where the Server is running (e.g., "DepartmentHost" or some such) or its IP address (e.g., "192.168.0.1"). Either you know it already or your System Administrator should be able to tell you. Whoever is responsible for running the Server should know all the right values you may need to use.

If this record is for a Local Server (which is not a real server at all, remember?), the Address value is meaningless, so you can leave it empty or type "N/A" or whatever in there.

Type/Port/Service is another important column. It is this column's value that determines whether this record is for a Local or "real" Remote Server. It can be set to several things. When set to the word "Local", it indicates that this is a Local Server. You can type it or pick it from the combo box. When set to anything else, it means this is a Remote Server. "Anything else" can be any of the following:

- Port Number. TCP/IP communications require the Address (and we've dealt with it earlier) and Port, which is simply a number. That number must be "right" for a particular Server running on a particular machine. Once again, someone like the System Administrator should tell you what Port Number to use here. The default is 12005.
- Blank. If the value in this column is left blank, it means "use default", which as we have mentioned is 12005.
- Service Name. That would be the name of the TCP/IP service that your copy of Windows knows how to resolve into a Port Number. Your System Administrator will tell you if you need to use something like this here and what the right value would be.

Admin Port/Service. Technically, it's very similar to Type/Port/Service described above. Except that this is the TCP/IP Port/Service used for Administration of the Server. When left blank, it means "use default", which in this case is 12006. If you are lucky enough to be this Server's administrator, you must ensure you have the correct value here – to be able to administer the Server, that is. Just accessing data from the Server does not require a proper value here, so if you aren't the server's administrator, you can safely leave it blank. Same goes for the Local Server.

Remote Encryption Password. This column contains exactly what its name suggests for the Remote Server. The Local Server doesn't support this, so just leave it blank. Every "real" Remote Server encrypts its communications with you, at least to some extent. Server Administration is a subject of another chapter. Once again, this is a value your System Administrator should provide you with, but we'll let you in on a secret: the default value is "elevatesoft". This password is per-Server: each Server has its own single password of this type.

Full Encryption. When checked, it tells Alventis to encrypt all communications with the Remote Server, including all data transfers. When unchecked, the only part that will get encrypted is the authentication (when Alventis sends your Username/Password to the server). Alventis uses Blowfish encryption, in case you want to know. The Server may be set-up to enforce full encryption, in which case your attempt to relax the security will simply be ignored. In general, encryption obviously slows things down a bit, but if you need it, that's a small price to pay for security.

Compression. This column specifies the Remote Compression level for communications with the Remote

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Server. You may want to use more or less compression depending on the speed of your connection to the Server. A 100Mbps LAN connection is not likely to benefit from any compression, for example. A slower Internet connection may go faster with compression level set to 5 or 6 – around the middle of the slider scale. You may have to experiment a bit to achieve the best results. Or just use what we suggested in the above examples: 0 for LAN / 6 for slower.

UName. This is your Username that you use to log into this Server.
For a Local Server, it can be any valid user known to it. This is described in detail in the chapter on Administration. For now, you should be happy to learn that if there are any users known to your Local Server, they appear in the combo box in this cell, so all you have to do is pick one.
For a Remote Server, you have to know what your Username for this Server is and correctly type it in the cell. No combo boxes to help you here, sorry.

Available. This is mostly an indicator column. A checked value indicates that you are properly connected to the server. An unchecked value means the connection could not be established. You can attempt to re-establish a connection to a currently un-Available server by checking this checkbox. If Alventis succeeds, the checkbox will get checked. If not, it will remain unchecked, so you can try later.

Enable. A Local Server is always Enabled, so the checkbox is grayed-out and you can't uncheck it. A Remote Server can be Enabled or not. When Enabled, it simply tells Alventis that you do want to attempt to connect to this Server. When disabled, Alventis won't even try. This is useful if you have one or more Servers in your list, and these Servers may not always be accessible. Whenever you launch Alventis, it tries to establish a connection to all Servers listed here. If the Server is accessible, this is very fast. If the Server is not accessible though, it takes Alventis several seconds (around 5 – per-server) before it gives up and marks the Server not Available (in the preceding column). If you know in advance that one or more Servers in your list are currently inaccessible (or you don't want to even try accessing them for some other reason), you can speed things up by telling Alventis to not attempt to access these Servers in vain.

Any Server modifications you make here take effect as soon as you post the corresponding record. For example, if a Remote Server was previously not Accessible to you because you had specified a wrong Username or Password, as soon as you post your new Username to the Servers grid, you will likely get prompted for your Password.

Creating a Server record creates just that: a record in a list. Likewise, deleting a Server record merely deletes the corresponding record in the list. Neither of these operations do anything to the Server.

A word of warning. Alventis keeps track of Servers by their SrvID number. If you edit a Server's record (which means its SrvID remains the same) everything will be fine, as long as that record keeps referring to the same Remote Server. You can freely edit the Name/Caption/Comment (nobody cares at all), as well as anything else you like – as long as this record is still "pointing" to the same Server. If however you edit it in such a way that it starts referring to a different Server, Alventis will likely get quite confused by this with unpredictable results. Editing of Address and Type/Port/Service would likely fall into this dangerous category. If this is what you absolutely want to do, proceed with great caution and: close all forms that are referencing anything from the Server you are about to modify (except the Database Explorer form of course). Once this is done, perform the editing you want and post your changes. Preferably, decline the connection to the new Server if prompted. Restart Alventis. A "fresh start" should let Alventis get properly "re-acquainted" with the modified Server. But your safest bet is to simply create a new Server record and, optionally, delete the old one.

There are 3 buttons floating on top of the Servers grid. They all act upon the currently-selected Server.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Administration opens the same-name form for the selected Server. This is a subject of the chapter on Administration.

Change Password allows you to modify your password for the specified Username used on the selected Remote Server. Local Servers don't use any password-protection, so for them, this button is disabled.

Forget Password is rather similar. It makes Alventis (just your local copy) forget your Password, if it remembered it in the first place. Hopefully, clicking this button won't make *you* forget your Password. Because you will be prompted for it next time Alventis needs to establish a connection to this Server. Which is the whole idea. This simply allows you to enhance your security that was previously somewhat relaxed by the fact that your Password was stored somewhere on your computer. See the Authentication topic for a bit more info.

Databases Grid.

This grid lists the Databases of the Server currently selected in the Servers grid. Such being the case, all Databases listed at any given time belong to the same (you got it: currently selected) Server. Consequently, just as Servers, Databases can be either Local or Remote. The only difference between how they are listed is in the Location column. We'll go over all the other columns first though.

DBID is as usual, simply a unique numeric ID for the particular Database record.

Created/Modified are the usual dates for the record.

Name/Caption/Comment can all be set to whatever helps you identify the Database. You should set at least the Name, but preferably all of them.

Location is a bit more interesting. For a Local Database, it specifies the file system path where the data tables physically reside. It can be an absolute path like C:\SomePath\MyData or a path relative to Alventis' program directory: just MoreData would put the Database in the subdirectory MoreData under whatever directory Alventis is installed in. You can specify UNC paths too. Something like \\ComputerName\Path. Check your Windows Help if in doubt.

For a Remote Database, Location specifies the logical name of a Database on the Remote Server. This is described in the chapter on Administration. Luckily, whenever Alventis successfully makes a connection to a Remote Server, it automatically lists here all its Databases, so all Locations will be appropriately filled out for you.

Manipulating Databases. As far as Remote Databases go, this is neither the place nor time. No, really: this is not the form where you manipulate these, and it is described in the chapter on Administration.

Local Databases may be created and deleted right from the Databases grid. There's really very little to it. To create a new Database, you simply create a new record in the Databases grid. The most important part is properly specifying the Location, and we already told you what it should be set to. To help you properly pick a directory accessible from your computer, you can click the little dotted button in the Location column and browse to the directory you want. If you want to create a new directory, just type its path in the box, and it will get created for you. When you post your new record, if the directory you chose does not already contain a valid Alventis Database, a new blank one is created for you. If you picked a directory of an existing Database, you just established a "link" to it, but no new physical data got created. That is, you just let Alventis know about that Database so that it could access it, that's all.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Deleting a Local Database record is – as is often the case in Alventis – less destructive than you may initially think. It merely removes the record, so Alventis "forgets" about that Database. It does not delete any data in the Database. Feel free to delete it manually, outside Alventis, using your favorite file manager – if this is what you really want to do. Alventis won't have the "evil deed" on it conscience though. This is for your data's own protection.

> Advanced and unsupported (i.e., "perform at your own risk") procedure: there's a useful consequence of what we have just explained. You can move a Database from one directory to another. The steps are: delete the Database in Alventis (the data remains intact); using your favorite file manager, move the data from its current location wherever you want; in the Database Explorer, create a new Database and point it to the new location of the data. An even more advanced unsupported procedure: you could move the Database from one directory to another without deleting/recreating the Database record. The procedure is quite similar: edit the Location to point to the new one; close Alventis; delete all newly-created data in the new location; move the data files from the old to the new location.

There are just a few more things you can do with Databases (regardless of whether they are Local or Remote) in this grid.
You can create a new UniGrid with the selected Database automatically added to it by simply double-clicking the Database you want.
You can add a Database to any currently visible UniGrid by dragging it onto the UniGrid's Tables Grid.

Finally, you can set a Default Database Style. Select the Database you want and apply a style to it like you would to a record in the Search Results. All tables of this Database (together with all their records) become painted with the style you chose. This is mostly helpful as a visual queue telling you at-a-glance the table or record's "origin".

2.12 Security Overview

This is an advanced subject. It is only applicable to the Enterprise version of Alventis. And it's just plain scary. So don't tell us we didn't warn you.

Alventis Multi-User Security Model

Alventis supports a rather sophisticated multi-user security model. The entire user/security system works on a per-server basis, so we'll assume that we're talking about a single server in the following discussion.

The users exist in a hierarchy. There exists a single "Root" user who should not be used by anybody (i.e., no copy of Alventis should be configured to access the server with Root credentials) for reasons that will become apparent shortly.

Every user is (or at least can be) a Manager. That means that other users can be his direct subordinates. Such direct subordinates of a user/manager form his Group.
It follows that each of these subordinates – being just a user at the same time – can also be a manager for more subordinates (his group), and so on.
This constitutes the hierarchical tree of users.
Direct subordinates of Root are said to be at the Level 1 of the tree. Their subordinates are at Level 2, and so on. You can have up to 50 levels, which should be enough for just about any enterprise.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Alventis' security revolves around Rights given to user Groups with regards to Objects. It's actually simpler than it sounds. Instead of giving you simplistic examples, we'll start by telling you what Objects we have in mind. They are:

- Table
- Form
- Select Query
- Non-Select Query
- Report
- Creation

The latter is not a real Object of course, but it's convenient to think of it as if it were one.

User Groups – we have already discussed. These are all users who are direct subordinate from one particular Manager user. It therefore makes sense to refer to that Group of users by simply referring to its Manager. Which means that in a sense – at least as far as Rights are concerned (which is all we care about) – a Group is essentially equivalent to its Manager. Not in the sense that the 3 users Bob, Tom, and Jim who report to their boss James are equivalent to him, no. It is in the sense that Bob does not have any Rights to any Objects on his own. Bob's access privileges are based on his Group's Rights, which simply mean: the Rights of James.

It's important to get this straight, so we'll try to say it differently. If we ever wanted to know what access privileges Tom has, we would not go looking for any Rights "attached" to his Username (those would be the rights of his Group of subordinates, but not of him). We would go "one level up" and see what Rights his Manager James has – because those would be the rights of James's group which includes Tom, and hence apply to him.

Each Right grants some access privileges (a few related ones). There is no mechanism per se in Alventis to deny an access privilege. Each Right applies on the one hand to a user (read: group/manager), and on the other – to an Object. (You can think of the Right as the central "bridging" element in a many-to-many relationship between users and Objects – if it helps and if you understand what it means).

Right Hierarchy.

Rights too have a hierarchy of their own. On the Object "side", a Right can apply to basically one of two things:

- An explicitly specified Object, e.g., Table Contacts of Database such-and-such.
- A class of Objects, e.g., Tables. This is typically expressed as "Default Table" and essentially means "all tables, unless there's a more specific Right to the table of interest".

On the user "side", there's even more choice:

- A particular Group of a particular user. Everything is specified explicitly, no deviations.
- A Level. A Right can be specified as applicable to all users who belong to a particular Level of the user tree (see above).
- Default. This simply means "everybody".

The overall principle behind all this is quite simple. Imagine a huge collection of various Rights. Some apply to Groups, some to Levels. Some apply to individual Objects, others to Object classes. Whenever Alventis must figure out if a specific user (the one currently logged into a specific Server from that specific copy of Alventis) has particular access privileges to a specific Object, it simply goes through the

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Rights collection in the following order:
    is there a Right for the specific Object and this user's Group
    is there a Right for the specific Object and this user's Level
    is there a Right for the specific Object and "Default" user
followed by:
    is there a Right for the Object class and this user's Group
    is there a Right for the Object class and this user's Level
    is there a Right for the Object class and "Default" user

As soon as Alventis finds a match (going top-to-bottom), it returns whatever access privileges that Right specifies. And yes, it just might be that somewhere down that list there does exist a Right granting even broader privileges, but Alventis won't take it into account. Incidentally, this can be seen as a way of "denying" someone some access privileges: Default user (i.e., everybody) may have full Rights to, say, a Table, but we could "give" one restricted user his own "personal" Right that does not grant any privileges to that same table. The more "specific" Right will be found first by Alventis – and that's what will be applied to this user.

Rights of Author.

And you thought it couldn't get any more complicated than that? Hang on.
Every Object in Alventis has an Author. This is just the user who created that Object, nothing fancy. But as we have discovered, Alventis users are arranged in a hierarchical tree. Wouldn't it be nice if this could be put to some twisted use? For example, make it so that a user could only access records created by members of his Group. Or a manager could access records of his subordinates but not his own manager. And so on. This is exactly the kind of flexibility that this feature offers you. Here's how it works.

Every Right has 3 Hierarchical Access Level specifiers: Read, Edit, Delete. Let's take Read, for example. The possible values are quite numerous, but they simply go in order restrictive-broader-broader-broader-... like so:

Self – means whatever access this Right is granting, it is only granted to Objects (say, records) whose Author is the user himself or any subordinate of his, even an indirect one. So, basically, he can read his own records plus those belonging to anybody "under" him in the tree/pyramid.

Group – means he is granted access to all records belonging to (i.e., created by a user of) this user's group plus all their subordinates. A little better.

Manager – means he gains access to records authored by his manager plus all of his manager's subordinates. Generally speaking, whenever a user gains access to records of user X, he is automatically granted same access to records of all X's subordinates. So we won't repeat it each and every time.

Manager's Group – access to records of his manager's entire group.

Mgr's Manager – higher up, we get to his manager's manager.

Mgr's Mgr Group – higher yet, the group of his manager's manager.

And so on – for all Levels ad nauseum. Essentially, the "movement" upwards in the user tree always works in steps: up to a manager; include his group; up to his manager; include his group, etc.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Hierarchical Access Levels make sense for some but not all types of Objects. They can be used for Tables, Queries of both kinds, and Reports. They are not appropriate (and are hence unavailable) for Forms (i.e., InfoViews) and Creation.
Use of Rights of Author / Hierarchical Access Levels may slow things down a bit in Alventis, but it's up to you to gage the performance drop. We do not expect it to be significant.

If you decide to steer clear from this feature because you might not need this much flexibility (or because we finally managed to confuse you out of your wits), you can effectively disable it by simply setting all (or some) Access Levels to a level which is higher than the height of your entire user tree. For instance, if your mighty user tree had 5 levels, it would be sufficient to set the Access Levels to "Mgr 6 up" which "overshoots" the Root of the tree and makes this whole feature irrelevant and thus disabled. To avoid counting levels at the risk of running out of fingers, you can simply set it to "Mgr 49 up" and forget it.

Before we get into all the nitty-gritty stuff, let's examine a scenario where Alventis Security could be used as a feature.

Human Resources. A company could have an Employees table that would record everything known about them. The HR department would want to have full access to all fields from the table. They could have an InfoView that implements all available fields and Queries/Reports to match. Managers may have a need to access just their subordinate's personal records, and perhaps not all the available fields. A different InfoView accessible to just them + appropriately set Hierarchical Access Levels could do the trick. Employees themselves may benefit from being able to access all records, but only the first/last name, telephone/extension, and e-mail fields. Yet another InfoView with just these fields would suit their needs. Since rights can be set not only on a per-table, but also on per-InfoView basis, it is possible to expose different sets of fields of the very same table to different groups of users.

This concludes the overview of Security in Alventis. Time to take a look at how all this is maintained and administered.

2.13 Security and Administration

This is an advanced subject only applicable to the Enterprise version of Alventis.
We are assuming that you have read the previous Security Overview chapter.

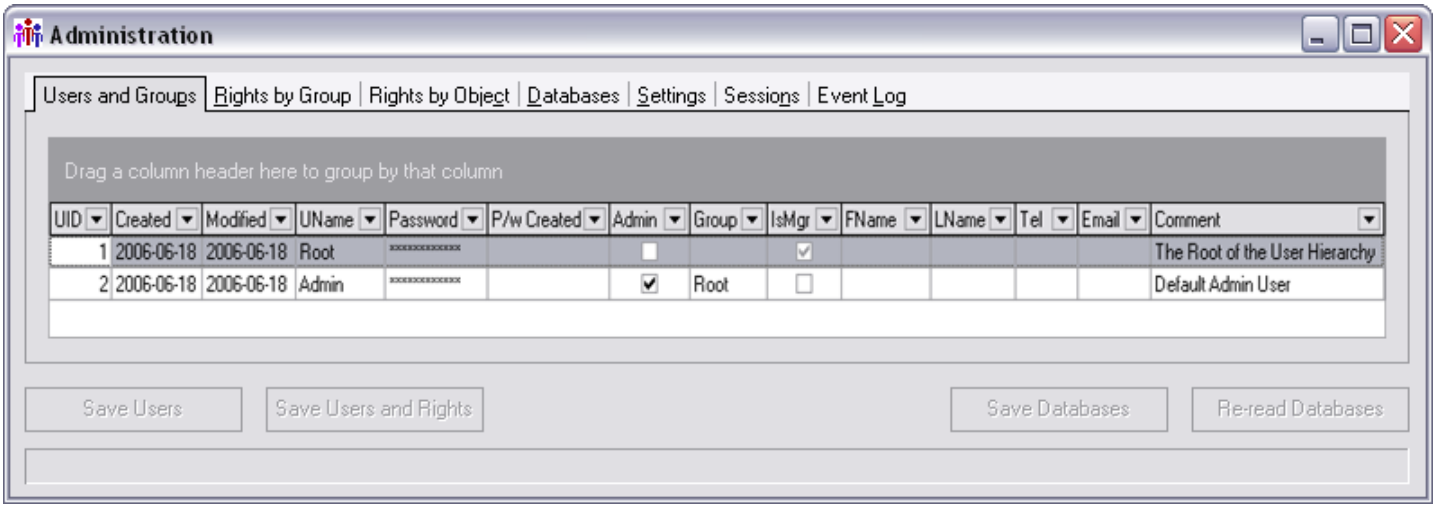
The Administration form is accessible from the Database Explorer form: highlight the Server you want to administer and click the Administration button. Note that everything that pertains to multi-user security in Alventis also applies to the Local Server – even if it is infinitely more useful for Remote Servers.

Now that Alventis' security model is intimately familiar to you, most of the Administration form itself should become quite obvious. The form uses a tabbed interface and has several buttons at the very bottom. The buttons pertain to some specific tab sheets, but have been placed here simply to remind you that their respective items may need saving (they will become enabled when this happens). Let's go over the tab sheets one-by-one.

Users

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The Users and Groups tab sheet has a single grid of users. Each user is a manager of his Group, remember? That's why there are no Groups per se listed anywhere. The only thing that pertains to user grouping here is the arrangement of the users in a tree structure by means of establishing the manager of every user. Except for Root who is at the very top of the tree and thus doesn't have a manager.



Let's do a quick run through the columns.

UID is the unique User ID. It is auto-assigned when a user record is created.

Created/Modified are the usual auto-set dates.

UName is the unique Username that you create to identify users. It will appear in InfoView and other forms when appropriate.

Password is the user's password. Passwords are case-sensitive and can include whatever characters you like. They can be up to 36 characters in length. All passwords are hidden behind asterisks to protect them from prying eyes. You can reveal and edit them by simply activating the corresponding cell, i.e., "entering" into it by hitting Enter or clicking the already-focused cell with the mouse. Can be useful when users inevitably forget their passwords.

P/w Created. Reserved for future use.

Admin specifies if a user has the ultimate privilege of administering this Server. There would normally be only a single user with this privilege, and this is preferred to keep security as tight as possible. But you may grant this privilege to more than one user if you wish. There's only one caveat: if more than one Admin users attempt to administer the same Server concurrently (and in particular, change the same "area": users, Rights, etc.), Alventis is likely to get confused. That is, the Admin who saves the configuration last is likely to overwrite the changes of the preceding Admin. So it is your responsibility to prevent this from happening. The easiest way is of course to have a single Admin. Otherwise, make sure they don't "step on each other's toes".

Group is the single column that establishes the tree hierarchy. It is here that you specify which manager's subordinate this user is. Just pick the right manager from the combo box. If an existing user does not appear in the list, check (in more senses than one) the next column.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

IsMgr is there to help you manage the user hierarchy. If you know that a particular user is not intended to become anybody's manager and have subordinate users, you can uncheck the IsMgr checkbox for him. All this does is remove his Username from the Group combo box's list, so it doesn't needlessly clutter it. If a user is already somebody's manager, the checkbox is grayed-out and you can't change it. If the user is currently nobody's manager, you can freely check/uncheck it.

FName/LName/Tel/Email/Comment. These are all for your information only. They are not used anywhere but here, so fill them out any way you find useful.

Once you modify anything in the user hierarchy, the Save Users button becomes enabled. You don't have to click it right away though: perhaps you want to adjust the user's Rights first to keep your security "in check". Just don't forget to save your changes at some point before closing the form, because Alventis will not prompt you to save your changes.

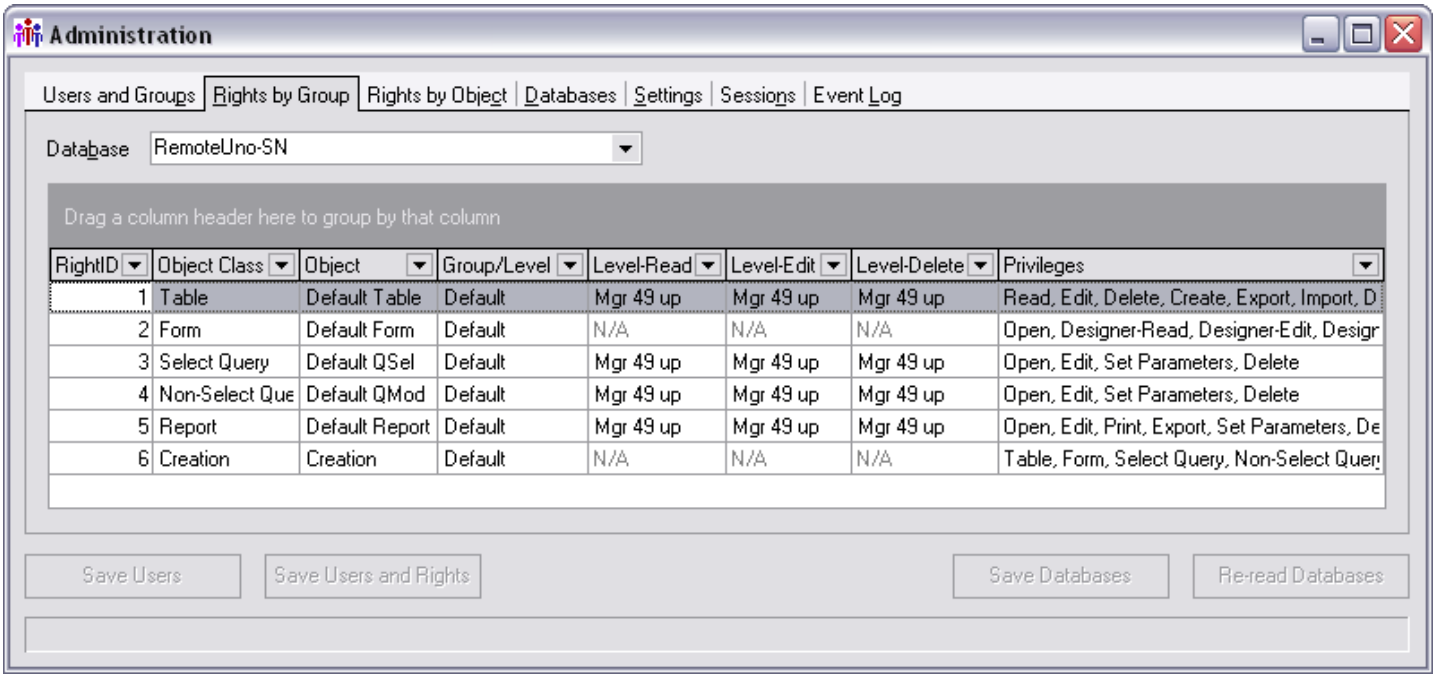
And that's all there is to maintaining users. Let's do something about their Rights now.

Rights

Sorting out who's got what Rights to what can get complicated. Which is why Alventis offers you not one but two approaches implemented in two separate tab sheets: Rights by Group and Rights by Object. Both contain the same exact information. It is only the way that information is presented to you that is different. Which one you prefer may depend on your personal taste or, more likely, the task at hand. You can freely switch between the two tab sheets as much as you like. In either case, Rights are listed on a per-Database basis. You can switch between the Server's Databases using the combo box at the top of either tab sheet. Doing so requires you to either save or abandon your changes (if any) to the currently-edited Database.

Rights by Group lists all Rights in a single "flat" grid. You can group the list by the Group/Level column to see what Rights each Group or Level has. This is what gives this tab sheet its name and makes it reminiscent of a Capability List that you may be familiar with. Let's go over the available columns.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Administration
Users and Groups
Rights by Group
Rights by Object
Databases
Settings
Sessions
Event Log
Database
RemoteUno-SN
Drag a column header here to group by that column
RightID
Object Class
Object
Group/Level
Level-Read
Level-Edit
Level-Delete
Privileges
1 Table Default Table Default Mgr 49 up Mgr 49 up Mgr 49 up Read, Edit, Delete, Create, Export, Import, D
2 Form Default Form Default N/A N/A N/A Open, Designer-Read, Designer-Edit, Desigr
3 Select Query Default QSel Default Mgr 49 up Mgr 49 up Mgr 49 up Open, Edit, Set Parameters, Delete
4 Non-Select Que Default QMod Default Mgr 49 up Mgr 49 up Mgr 49 up Open, Edit, Set Parameters, Delete
5 Report Default Report Default Mgr 49 up Mgr 49 up Mgr 49 up Open, Edit, Print, Export, Set Parameters, De
6 Creation Creation Default N/A N/A N/A Table, Form, Select Query, Non-Select Quer
Save Users
Save Users and Rights
Save Databases
Re-read Databases

RightID is as usual the unique auto-incrementing numeric ID auto-assigned to each Right when it is created.

Object Class is the Object Class: Table, Form, Select Query, Non-Select Query, Report, or Creation.

Object displays the Object this Right applies to. As we have learned earlier, it can be, e.g., a specific explicitly-specified table such as Contacts or "Default Table". Same goes for most other Object Classes.

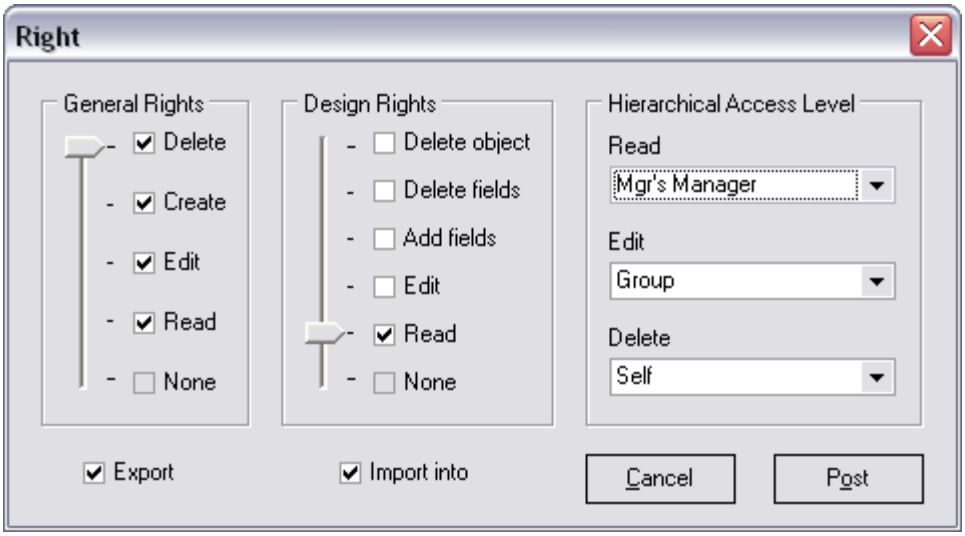
Group/Level indicates who this Right applies to. This can be "Default", a particular Level, or a particular Group.

Level-Read/Level-Edit/Level-Delete are the Hierarchical Access Levels for this Right.

Privileges is a column that displays the list of access privileges this Right grants.

Only the 3 Hierarchical Access Level columns allow direct editing right in the grid. The rest of them are read-only. To edit the Right, you double-click it in the grid. This will open the individual Right in a dialog box. The appearance of the dialog box depends on the Object Class whose Right you are opening. While the specific contents of the dialog may vary slightly, the basic principle is always the same, so we'll only illustrate it on the example of a Table Right dialog.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



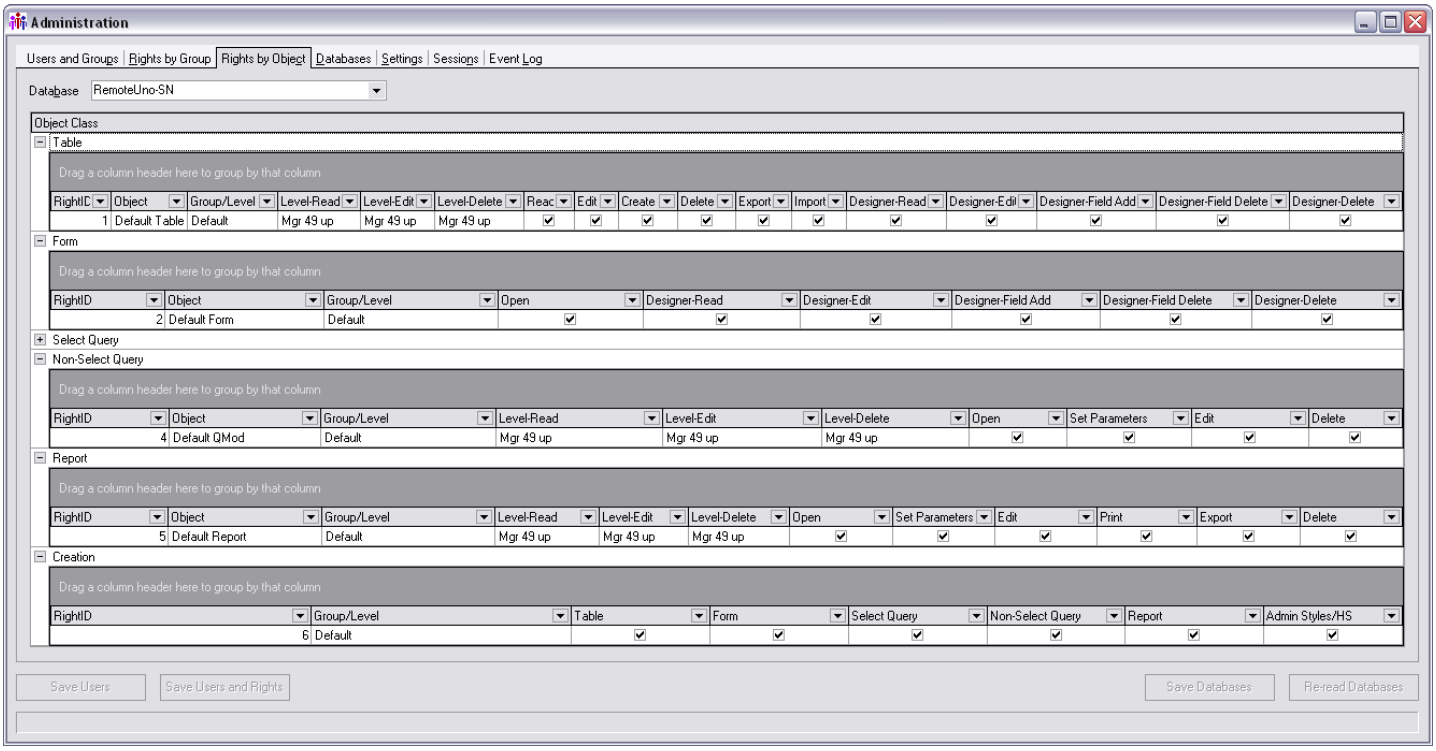
All the checkboxes represent individual access privileges. Those that are grouped together with a Slider control have a certain degree of inter-dependency: enabling a certain privilege automatically enables "lesser" privileges and vice versa. Higher privileges are higher on the Slider scale, so it should be quite intuitive. You can use either the Slider or the checkboxes (or both).

The Hierarchical Access Level group will be present for Rights to which it applies, and there should be nothing enigmatic about it by now.

We'll discuss the specific access privileges of each Object Class shortly.

Rights by Object lists the very same Rights, but in a hierarchical grid or set of grids grouped by Object Class. As such, they can be seen as similar to an Access Control List you may have encountered elsewhere.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



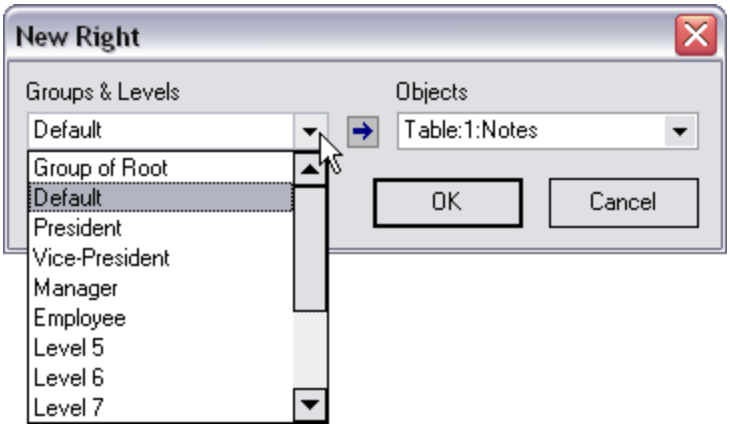
The first several columns are identical to those of the Rights by Group grid described above, so we won't repeat ourselves.

The remaining checkbox columns represent individual access privileges. As simple as that.

In this combined grid, you can edit Rights directly by altering values of the corresponding cells. You can still open each Right in its own dialog if you prefer – again, by double-clicking a Right in the grid.

Whichever Rights grid you use, you can create a new Right as if you were creating a new record in the grid – by hitting the Insert key. You are prompted for what the new Right should apply to and who it should apply to, i.e., the Object and the Group/Level respectively. The prompt automatically ensures that you do not create duplicate Rights, i.e., two Right records that apply to the same pair of what/who. Once a Right "connecting" an Object and a Group/Level has been created, you cannot modify this connection information. You can only edit the Right's access privileges and Hierarchical Access Levels, if applicable. You can of course simply "replace" a Right by deleting the old one and creating a new one. You can delete a Right from either grid as you would delete any grid record: highlight the Right and hit Delete or Ctrl-Delete on the keyboard.

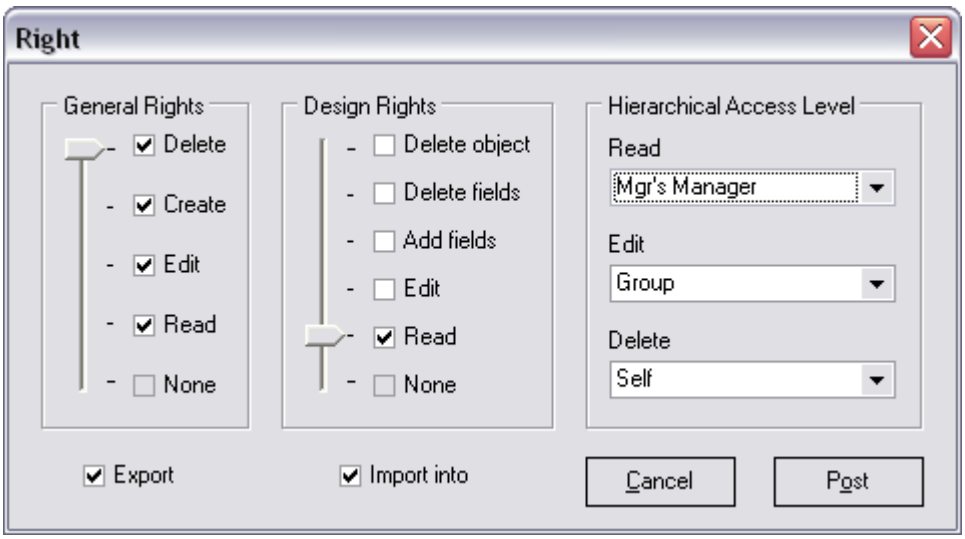
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Whenever you create a new Database, Alventis initializes Rights to a "maximum access" default set that gives everybody full Privileges.

Let's quickly go over the specific Access Privileges that you can grant to Objects of specific Classes.

Tables have the largest assortment which is listed below.



General Rights pertain for the most part to privileges that affect your interaction with tables and their records in Alventis:

- Delete means right to delete records in the table
- Create means create new records in the table
- Edit means edit existing records
- Read means, well, read or view records

Design Rights are related to what you do with a table in Alventis Designer:

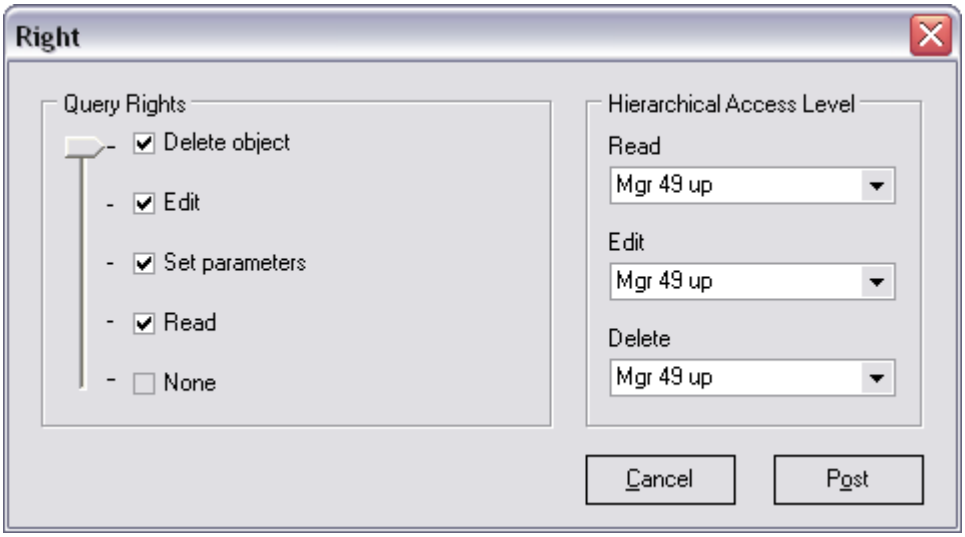
- Delete object means delete the table itself
- Delete fields means delete any of its fields
- Add fields means create new fields
- Edit means edit existing fields
- Read means gain at least read-only access to a table

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Besides the above grouped privileges, there are two more:
Export means export the table or its data from Alventis
Import into means import data into the table

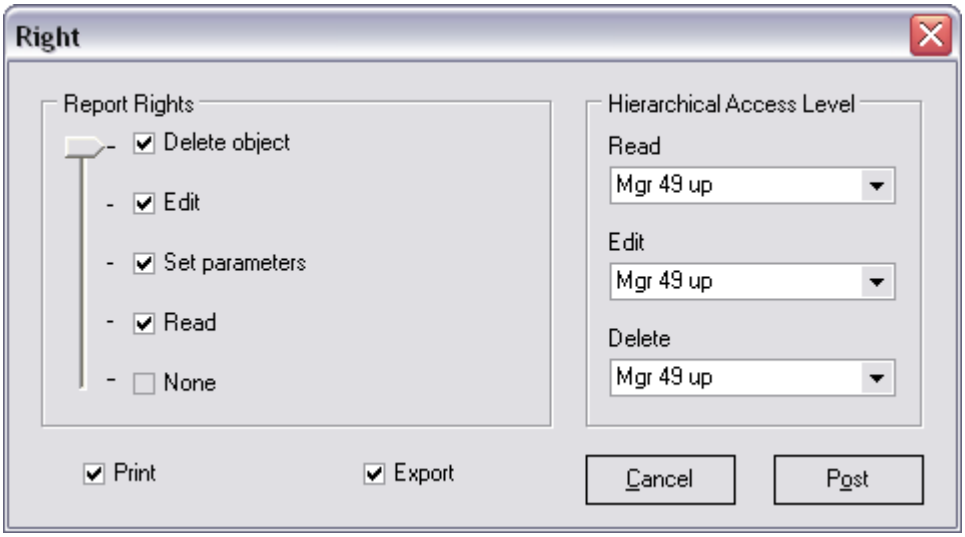
Forms have a group of Design Rights that greatly resemble those of tables:
Delete object means delete the form itself
Delete fields means delete any of its fields
Add fields means create new fields
Edit means edit the form in any way
Read means being able to at least open the form in Designer

Beyond these, there's a single right relevant to forms in Alventis:
Open means opening the InfoView form in Alventis



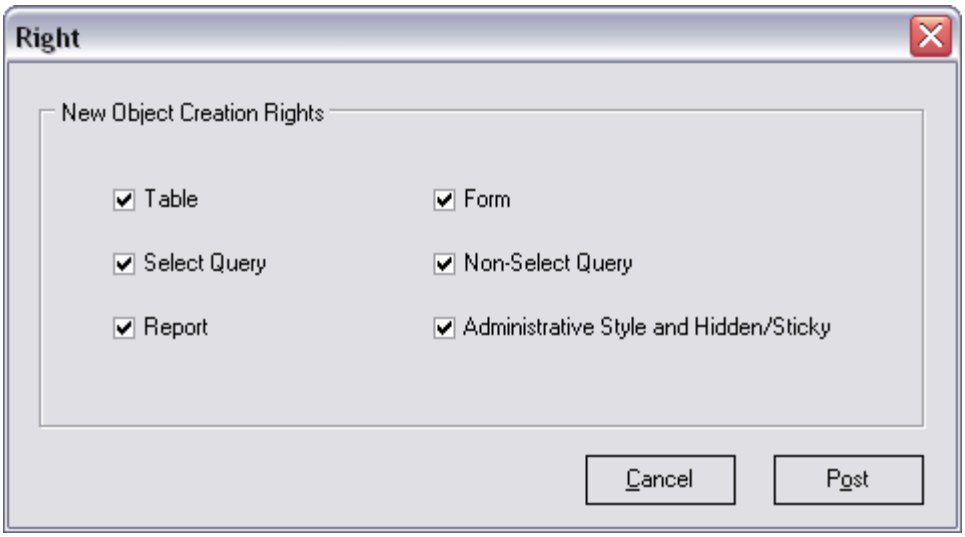
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Both Select and Non-Select Queries have the same set of privileges:
- Delete object means delete the query
- Edit means edit the query
- Set parameters means editing the query parameters
- Read means opening the query and executing it



Reports have an identical set of privileges with a virtually identical meaning:
- Delete object means delete the report
- Edit means edit the report
- Set parameters means editing the report parameters
- Read means opening and using the report to produce output

In addition to the above, Reports have two more:
- Print is rather obvious
- Export means exporting Report Output via the Save Output command



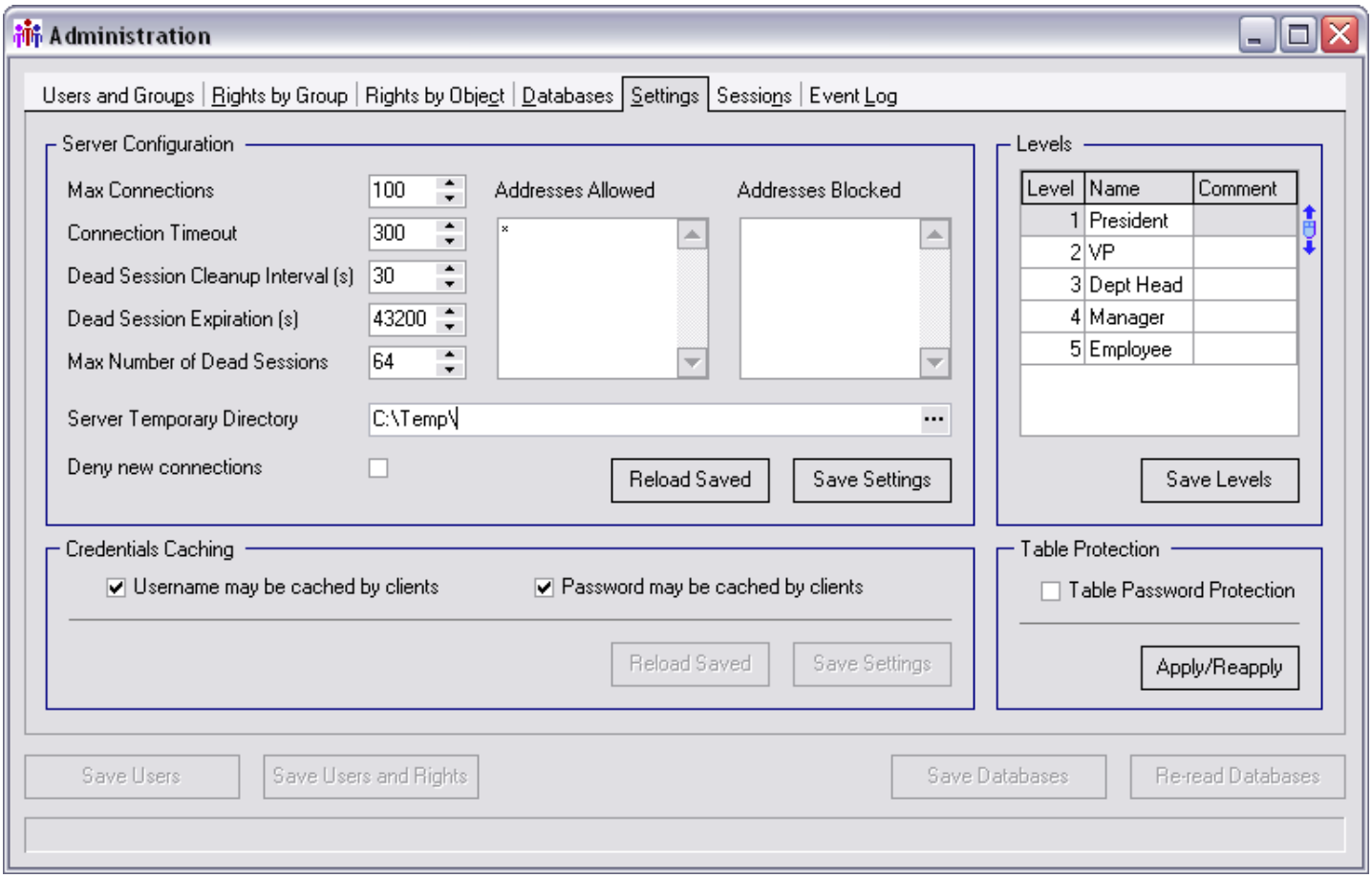
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Creation has privileges to create new objects in either Alventis or Designer, as the case may be. We'll just list them for the sake of completeness:

- Table
- Form
- Select Query
- Non-Select Query
- Report
- Administrative Style and Hidden/Sticky

Once you have made any changes to Rights, the Save Users and Rights button becomes enabled. As its caption suggests, it saves Users/Groups if necessary, and their updated Rights (it is not possible to save Rights updates without saving the users first). Don't forget to press this button before exiting the form – if you do indeed want to save your changes.

The remaining tab sheets pertain to the administration of the Server but have almost nothing to do with security, except for a portion of the Settings tab sheet, so let's get it over with.



The part that interests us is on the right-hand side of the tab sheet. At the top, we have the Levels grid. This allows you to essentially name the Levels the way you want to make them more identifiable in whatever Right-related combo boxes list them. The Level column is auto-set to the level number. The Name is the level name that will appear in combo boxes. The Comment is as usual, for your "notes to

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

self". You have to name levels in sequence, i.e., you can't have named levels 1 and 2 and suddenly decide that you would like to name level 5: you have to name levels 3 and 4 first. You can re-order existing levels by dragging them around with the mouse, even several at the same time.

The Table Protection group has just a single checkbox Table Password Protection and an Apply/Reapply button.

But before we tell you what this checkbox does, let's discuss the issue of data file security for a moment. In this case, we are no longer talking about Alventis' Security Model, Rights, Groups, and so on. All of the above can allow or prevent Alventis from accessing certain records or kinds of items. The key word in the last sentence is "Alventis". Alventis takes upon itself the responsibility of accessing objects according to its user's privileges. Any other application capable of accessing the data would not be bound by this responsibility and would therefore gain full unrestricted access. But it's not quite that simple either. Let's see what happens between a Client (Alventis in our case – but generally speaking any application that accesses the data from the Server) and a Server when they communicate.

First, the Client needs to know the Server's address to even attempt to establish a connection. In the case of the Alventis Server, this corresponds to Address and Type/Port/Service. Plus the Client needs to know the Server's Remote Encryption Password. Unlike the Address and the Port, the latter is not directly visible to Alventis users, so a security-conscious System Administrator could install Alventis for everybody in the enterprise without ever telling them what the Remote Encryption Password is. Alventis would know it but the user would not. That's not the only password at play though. Every user also has his personal password – the one he uses in the Authentication Dialog. So, an "outsider" who may have found out the Server's Remote Encryption Password still needs a valid Username and Password to make a successful connection to the Server. This means we only need to worry about a malicious user with valid credentials and the knowledge of the Remote Encryption Password. If a mean-spirited user did know both of the above passwords, he could then, in theory, connect to the Server using an application other than Alventis and gain direct access to whatever tables the Server has to offer. We are not aware of any applications that would provide such functionality, so it's a somewhat far-fetched scenario, but it is possible. We can imagine yet another one. If the Server (the application that constitutes the Alventis Server) is running on a machine that someone can gain direct access to (physically or via the network), he could obviously get to the data table files and do whatever he wants with them.

The solution to the last problem is: run the Server under properly configured Windows on a well-secured machine that users would have no physical or other unauthorized access to. The only open access channel would be the Port needed by Alventis. Even a partial discussion of this level of security is beyond the scope of this Guide, so you'll have to refer to any good book on Security. But we still haven't done anything about the possibility of someone using the proper Port access channel from software other than Alventis. Well, there's really only one thing you can do against this hypothetical threat: encrypt all data tables (and some system tables used by Alventis).

The tables are encrypted on-disk using the strong Blowfish algorithm, so the perpetrator would have to know yet another password regardless of the fashion in which he may have gained access to the Server (physical or via something other than Alventis). The good news about that Table Password is that neither the Alventis user, nor the System Administrator, nor even we know what the password is, and we'll just leave it at that. Alventis and the Server arrange it between themselves. The bad news is... well... the same: since nobody knows the password, if something were to go wrong (data corruption caused by power outage or hardware or software failure) you put yourself at a somewhat greater risk of losing access to your data. On the other hand, if you are this serious about data security, you are undoubtedly serious about data backups, right? So, if you are in a habit of performing regular rotating data backups, the risk of losing your data can hopefully be considered negligible. You should still

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

remember that it is your full responsibility to prevent data loss by using backups, uninterruptible power supplies, reliable hardware, and a well-tuned installation of Windows. Oh, did we mention that you should do regular backups?

So, now you know. If you check the Table Password Protection checkbox and hit Apply/Reapply, the Server's tables become encrypted on the Server's disk. You can leave the checkbox checked (assuming it already was in that state) and hit Apply/Reapply again, in which case it will re-encrypt everything with a different password. To decrypt the tables, uncheck the checkbox and – guess what – hit the Apply/Reapply button.

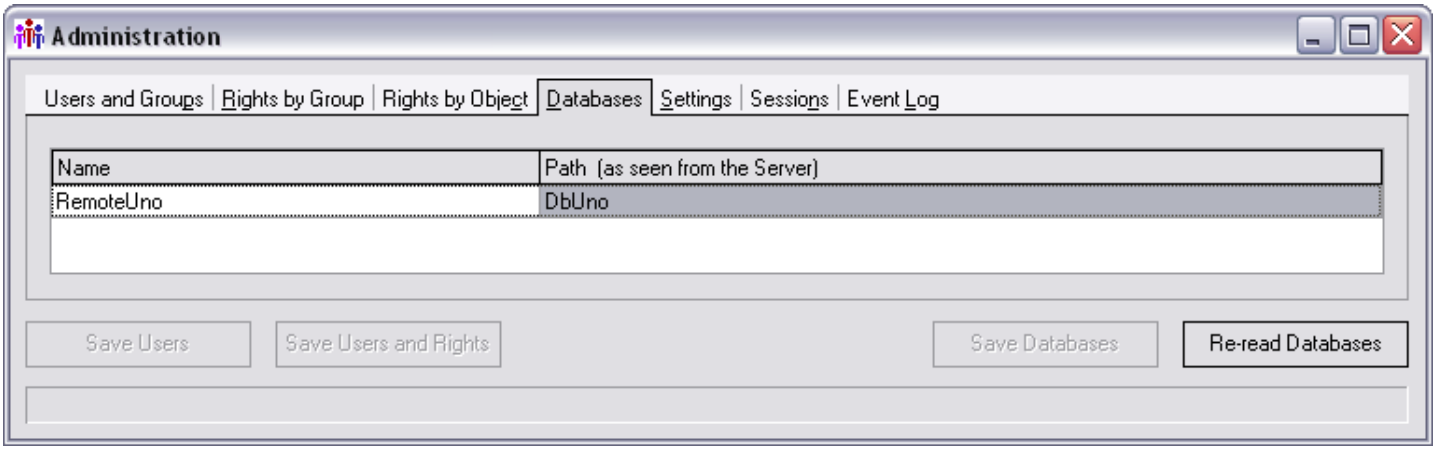
This concludes the topic on Alventis Security Model and Implementation.

2.14 Server Administration

This chapter is only relevant to Remote Servers. Local Server doesn't have any settings we are going to discuss here.
Aside from Database maintenance, if you insist, which for the Local Server is performed directly in the Database Explorer form.

Server Administration is performed in the 4 last tab sheets of the Administration form. The first 3 tab sheets are dedicated to Alventis Security and have been dealt with in the Security and Administration chapter.

The Databases tab sheet is remarkably simple.



Name is the name of the Remote Database. This is what appears in the Location column in the Databases grid of the Database Explorer form for Remote Databases. You can't change it for an existing Database.

Path is the physical file system path to the directory on the Server where the Database's data tables are located. This path is specified relative to the Server application. That is, it is the Server that will be resolving it from its "point of view". Let's illustrate this with an example. Imagine we have the Server running on a computer called Alpha, which has (or will end up having) a Database in the directory C:\ServerData. Imagine that were are administering this Server from Alventis running on another

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

computer called Theta. Imagine finally that Theta has direct access to Alpha's hard disk C: - either via a mapped drive letter R: or using a UNC path. You may be tempted to enter the path to the Database directory as you may see it from Theta: either as R:\ServerData or as \\Alpha\C$\ServerData. The latter may even work, but you shouldn't use it, and the former certainly won't work at all. The path that should be entered is C:\ServerData – because this is what the Server running on Alpha sees and understands.

To further confuse you, there's a little dotted button in the Path cell that helps you browse to a directory. It will work great if you are administering a Server running on the same machine you are administering it from since in this case both Alventis and the Server are sharing a common "point of view" of the file system. If administering a Server on a remote machine though, you must use this button with great caution. Even if you do manage to browse to the remote directory you want, once you have done so you must edit the beginning of the path to make it "understandable" to the Server (as in the example above where we replaced the beginning by "C:").

Any modifications enable the Save Databases button. As with other similar buttons in this form, you must remember to save your changes before closing the form (it won't remind you).

The Re-read Databases button abandons your changes, if any, and updates the list with whatever databases currently exist on the Server.

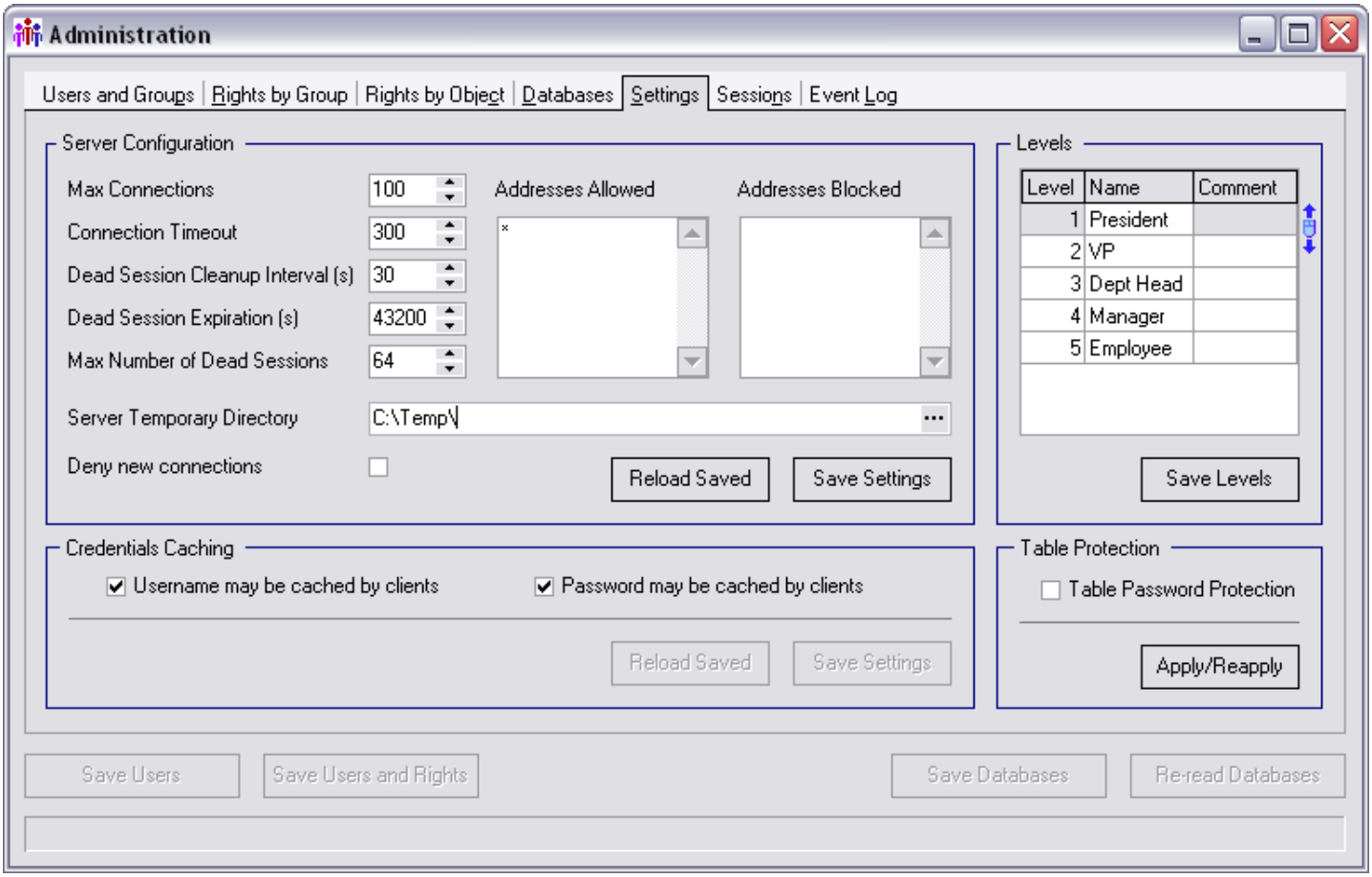
Editing the Path of an existing Database will not move any existing data to the new location. It will create a new blank database there and abandon the old.

Deleting a Database does not delete any data tables either. It merely removes the Database in question from the Server's internal list of Databases known to it.

If you do need to "move" your Database, you may want to simply copy its entire data directory to the new destination (using your favorite file manager – and in all likelihood when the Server is shut down and nobody is connected to the Database), and only then change the Path in the Databases grid. Once this is done, you can remove the original directory at your leisure.

The Settings tab sheet is depicted below.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



We have already been through the Levels and Table Protection groups of controls on the right in the chapter on Security, so we will only describe here the controls we find on the left.

Server Configuration group contains everything related to the Server's setup that can be configured remotely.

Max Connections specifies the maximum number of concurrent connection the Server may accept. The default value is 100. The maximum possible value is 32,765. You may want to set this value to some low number if you want to limit the number of concurrent connections for performance or other reasons.

Connection Timeout specifies for how many seconds a Session may remain Idle before the Server automatically disconnects it. Default: 300.

Dead Session Cleanup Interval specifies, in seconds, how often the Server checks for expired Dead Sessions. Default: 30.

Dead Session Expiration specifies after how many seconds a disconnected Session is considered Dead. A value of zero (0) indicates that Sessions are never considered Dead. Default: 43200 (12 hours).

Max Number of Dead Sessions specifies how many Dead Sessions the Server will keep around before it starts removing them in the oldest-first order. Default: 64.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

There's little reason for you to care about it, but let us briefly tell you in case you do. Alventis communicates with a Server using what is known as a Session. A Session is established when Alventis makes its first request to the Server, and may persist indefinitely – depending on Alventis' activity. While Alventis is actively communicating with the Server, the corresponding Session is active. As soon as Alventis makes a pause, the Session becomes Idle. If the Session remains Idle for a period that exceeds the Connection Timeout value, the Session is disconnected. If after that the Dead Session Expiration time interval passes (if it is non-zero), the Session is considered Dead. The Server periodically removes Dead Sessions – once every Dead Session Cleanup Interval seconds. If Dead Session Expiration timeout is set to zero, disconnected Sessions do not become Dead, but the Server simply removes them when their number reaches Max Number of Dead Sessions.

You are not likely to ever need to modify the above settings.

The Addresses Allowed list box is where you can specify Allowed IP addresses. Default: "*", i.e., "everybody".

The Addresses Blocked list box lists IP addresses from which the Server will not accept any connections. Default: empty list.

The IP addresses can be specified explicitly or using the asterisk wildcard, e.g., "192.168.5.*".
Be careful to not deny yourself access to the Server!

Server Temporary Directory is the directory, on the Server's machine, where it stores any temporary files. It defaults to the Server's %TEMP% directory. This directory must be specified from the Server's "point of view" just like the Path of a Remote Database – see the discussion pertaining to that.

The Deny New Connections checkbox, if checked, will make the Server not accept any new connections.

The Reload Saved button abandons your changes, if any, and reloads the current configuration from the Server.
The Save Settings button updates the Server's configuration.

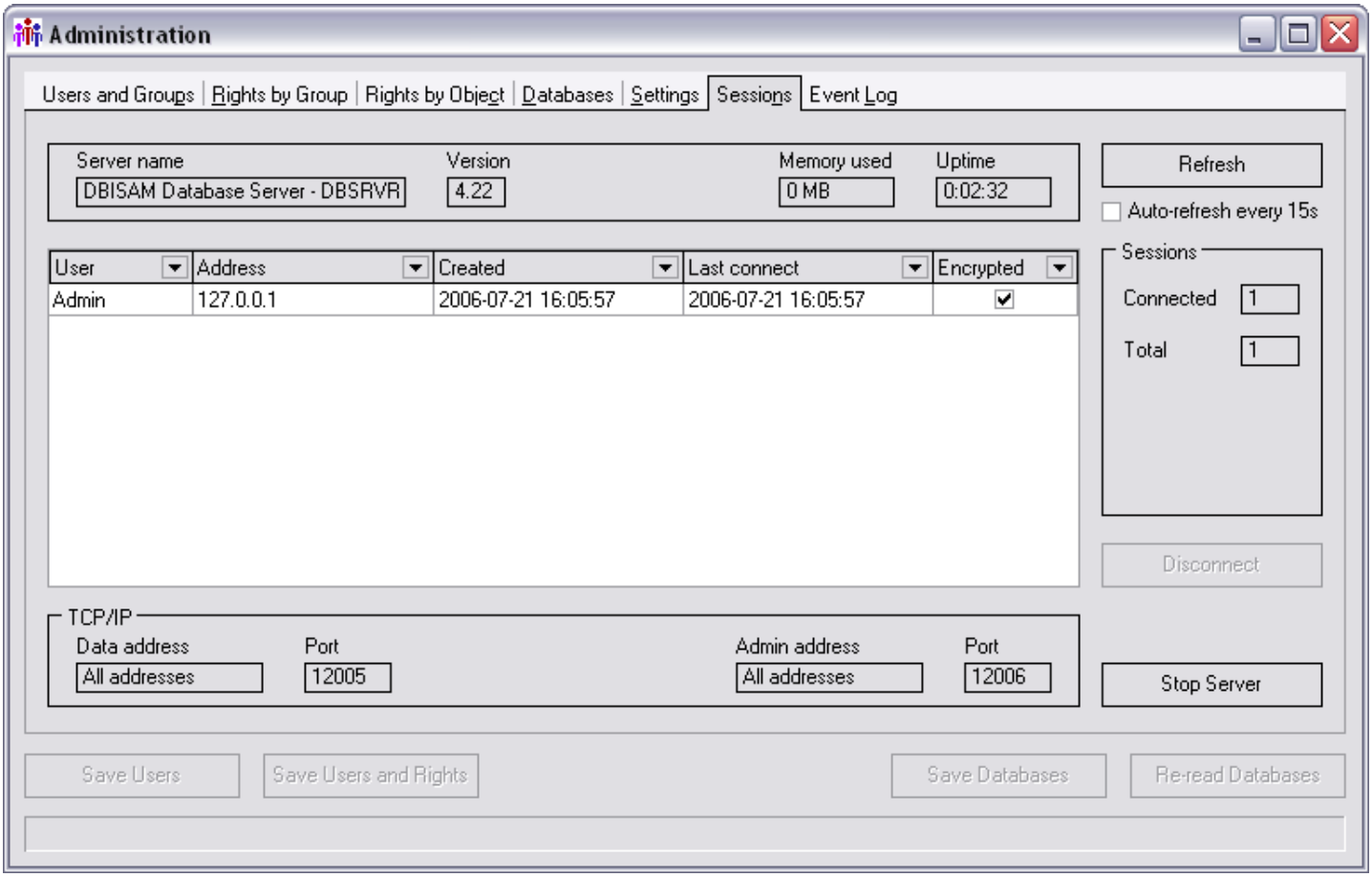
The Credentials Caching group of controls specifies how lazy this Server's users are allowed to get.
Username may be cached by clients, when checked, means that if a user wishes to do so, he can let his copy of Alventis (his "client" software) remember his Username. To do so, he'd check the corresponding checkbox in the Authentication dialog.
Password may be cached by clients does the same exact thing – but for the user's Password.

Enabling both checkboxes relaxes your security a bit since now each user's local copy of Alventis will have his credentials stored somewhere (even though a user is not likely to be able to read them).

You can disable Username and/or Password caching at any time. The only issue you should be aware of is that Alventis will still use the cached credentials (if it has already cached them in the past) at the time of the next login. At that point, Alventis will diligently "forget" the cached value to comply with the new Server policy, so subsequent logins will require the user to enter his credentials by hand.

The Sessions tab sheet may look busy, but it mostly presents read-only status information.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



At the very top we find some Server status information boxes: Server Name and Version, as well as Memory used by the Server application and its Uptime.
At the bottom there are a few TCP/IP settings: the Data Address and Port pair and the same for the Administrative access.

The chapter on the Server Installation explains how these are set.

The middle of the tab sheet is occupied by a list of Sessions that currently exist on the Server.
User shows the Username this Session belongs to.
Address will display the IP address the Session is connected from. If the Session is Dead (i.e., it has been Idle long enough), this column will change to "Was " followed by the last-known IP address.
Created is the time the Session got created.
Last connect is the last time the Session (re)established a connection.
Encrypted tells us if the Session used full data encryption, which would be the case if the user enabled the Full Encryption checkbox for this Server.

To the right of the Session list, two status boxes display the Total number of Sessions and how many of these are currently Connected.

The Disconnect button just below allows you to manually disconnect the selected Session. To be perfectly honest, we can't think of any reason for you to do so.

In the top/right corner there's a Refresh button that allows you to manually refresh all the info

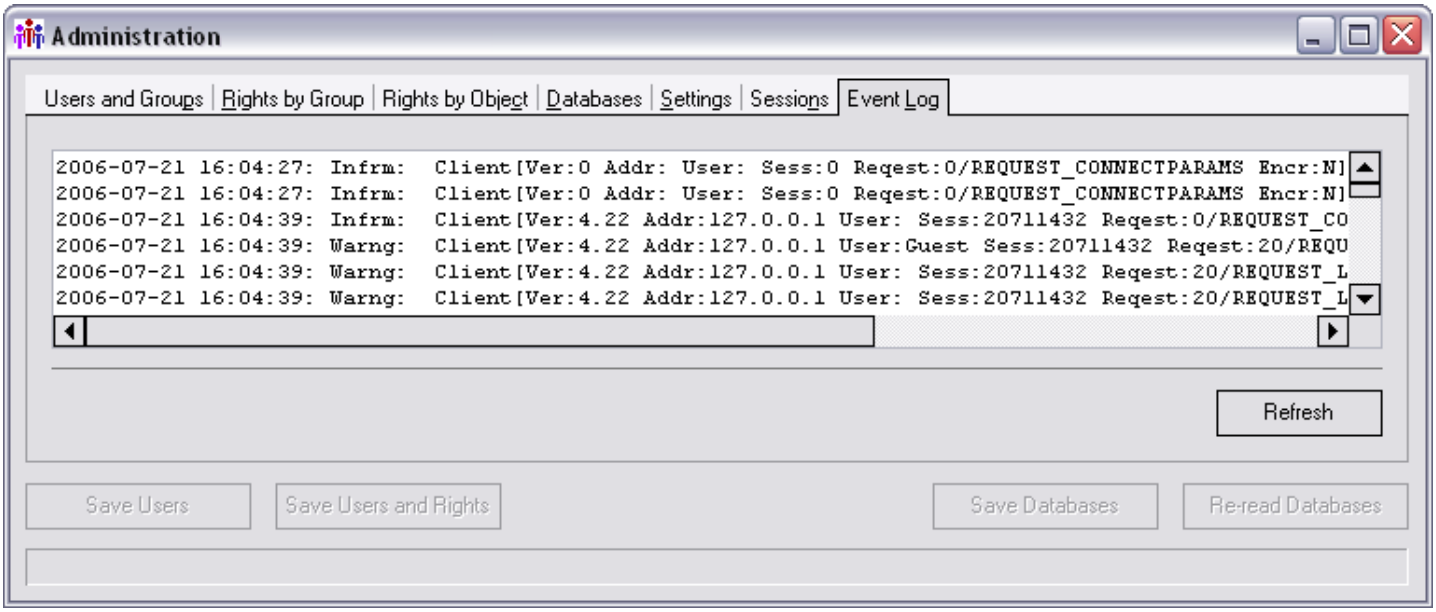
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

displayed by this tab sheet. You can force it to auto-refresh every 15 seconds by checking the checkbox under the button.

Lurking in the bottom/right corner is the Stop Server button. It allows you to stop (and restart) the current Server. Stopping the Server will close all active connections, which may lead to users losing their work if they are in the middle of an unposted editing operation, so be very careful.

The Event Log tab sheet displays the current event log of the Server. It uses the following format: Date/Time, Event Type (which could be: Infrm/Warng/Error which stands for Information/Warning/Error respectively), some Client information in brackets: Version, Address, Username (if available), Session ID, Request, and whether it was Encrypted, and all this is followed by the Status Message, e.g., "Connection accepted".

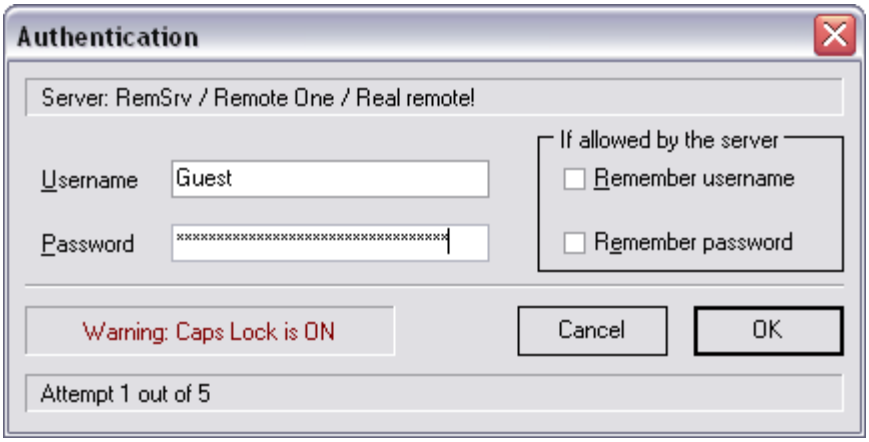
The Refresh button at the bottom re-reads the log from the Server.


Administration
Users and Groups
Rights by Group
Rights by Object
Databases
Settings
Sessions
Event Log
2006-07-21 16:04:27: Infrm: Client[Ver:0 Addr: User: Sess:0 Reqest:0/REQUEST_CONNECTPARAMS Encr:N]
2006-07-21 16:04:27: Infrm: Client[Ver:0 Addr: User: Sess:0 Reqest:0/REQUEST_CONNECTPARAMS Encr:N]
2006-07-21 16:04:39: Infrm: Client[Ver:4.22 Addr:127.0.0.1 User: Sess:20711432 Reqest:0/REQUEST_CO
2006-07-21 16:04:39: Warng: Client[Ver:4.22 Addr:127.0.0.1 User:Guest Sess:20711432 Reqest:20/REQU
2006-07-21 16:04:39: Warng: Client[Ver:4.22 Addr:127.0.0.1 User: Sess:20711432 Reqest:20/REQUEST_L
2006-07-21 16:04:39: Warng: Client[Ver:4.22 Addr:127.0.0.1 User: Sess:20711432 Reqest:20/REQUEST_L
Refresh
Save Users
Save Users and Rights
Save Databases
Re-read Databases

2.15 Authentication Dialog

Accessing a Remote Server requires you to "present your credentials" consisting of your Username and Password. If Alventis does not already know them, it will present you with the opportunity to specify them in the **Authentication Dialog**.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



You are given 5 attempts to properly specify a valid Username and Password combination. The Password is case-sensitive.

The two checkboxes on the right allow you to ask Alventis to remember your Username and, optionally, Password so as to not prompt you in the future. Having Alventis remember both obviously lowers your security if you are not the only person who has access to your computer. Knowing this, a security-conscious Server (well, its administrator, of course) may not allow your copy of Alventis to remember either or both. This is controlled from the Credentials Caching group of controls in the Settings tab sheet of the Administration form. You cannot know in advance what the Server may permit you to remember, and Alventis will ignore your request if it has to, so don't be surprised if it "doesn't work": Alventis may be simply complying with the Server's security policy.

You can Cancel the dialog thus abandoning your attempts to connect to the Server. This would happen if you got prompted for your credentials when you did not really intend to make a connection or if you realized that you don't remember your Username or Password.

2.16 Queries List

This chapter contains some fairly advanced subject matter. However, even if you don't know what a query is and don't intend to ever create one, you will still interact with default queries created for you by Alventis, so don't get too intimidated.

Alventis supports queries using a broad subset of the ANSI SQL-92 specification. Queries can be written by you and executed against the data tables of the Database the queries belong to. A large number of query types is supported, such as the following:

- SELECT
- UPDATE
- INSERT

and many more. Select queries are considered to be in one category of queries since they do not modify any data and produce a set of record as "output". All other types of queries are considered to be in the "Non-Select" category.

Alventis also supports Reports as a means of arranging selected records and formatting their contents for printing or some other form of output. Every Report is based on the output of a Select query. Non-Select queries don't produce any output, and so cannot be a source of a Report.

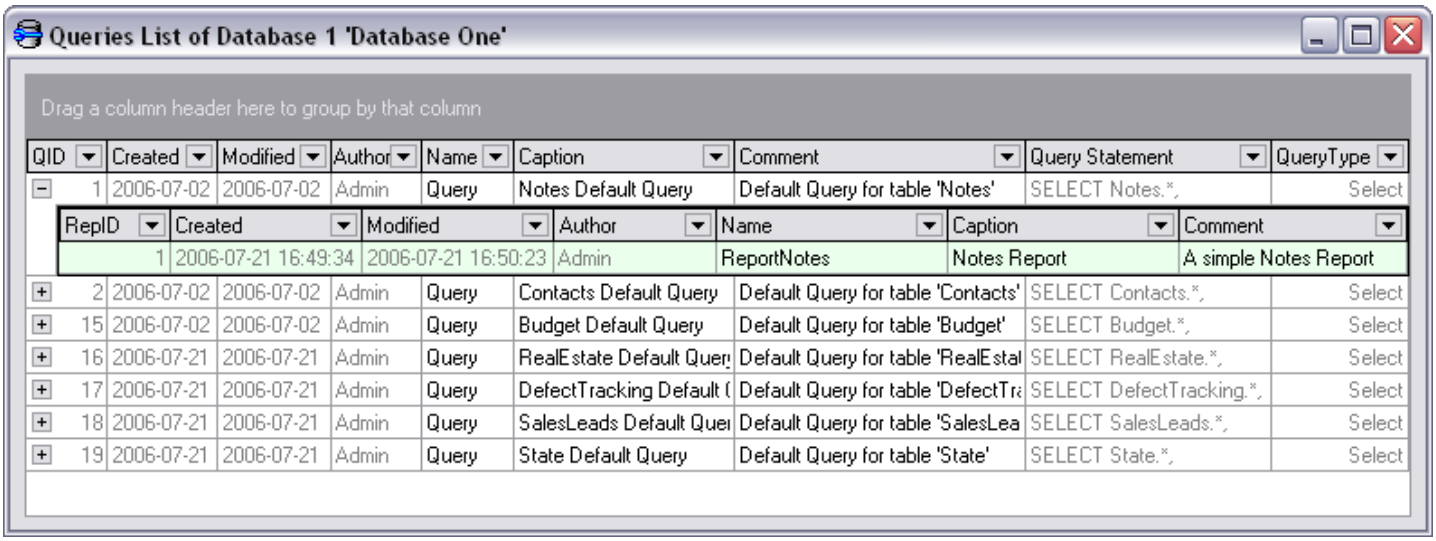
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

You may not know (or ever want to know) how to write Queries, but you will in all likelihood want to produce Reports. This is why Alventis makes your life easier by supplying a Default query for each and every data table. The Default query is nothing more than a rather simple SELECT query mainly based on the data table it "represents". The only twist to it is that the Default queries look-up the Username of the Author of each record. For now though, all you need to know is that for every table you may have, there is at least one Default query based on it, that allows you to base a Report on it.

There may be any number of queries. In the simplest case, a query may be based on a single table, the simplest possible such query being SELECT * FROM Contacts, for example. A query may use more than one table as its source though, so in general, it makes little or no sense thinking of a query being somehow "attached" to some specific table. Which is why what you have in Alventis is simply a "flat" list of queries. There is one such list per Database.

There may also be any number of Reports. Each Report is based on one and only one Select query, so in this case it does make sense to say: you can have one or more Reports per query. You can also have a query with no Reports based on it, of course.

The Queries List form is then quite simple. It lists all queries of the Database, and under each query that has Reports based on it – the Reports. You end up with a two-level grid, where queries occupy the top level, and Reports – the second level "under" their queries.


Queries List of Database 1 'Database One'
Drag a column header here to group by that column
QID | Created | Modified | Author | Name | Caption | Comment | Query Statement | QueryType
1 | 2006-07-02 | 2006-07-02 | Admin | Query | Notes Default Query | Default Query for table 'Notes' | SELECT Notes.*, | Select
RepID | Created | Modified | Author | Name | Caption | Comment
1 | 2006-07-21 16:49:34 | 2006-07-21 16:50:23 | Admin | ReportNotes | Notes Report | A simple Notes Report
2 | 2006-07-02 | 2006-07-02 | Admin | Query | Contacts Default Query | Default Query for table 'Contacts' | SELECT Contacts.*, | Select
15 | 2006-07-02 | 2006-07-02 | Admin | Query | Budget Default Query | Default Query for table 'Budget' | SELECT Budget.*, | Select
16 | 2006-07-21 | 2006-07-21 | Admin | Query | RealEstate Default Quer | Default Query for table 'RealEstal | SELECT RealEstate.*, | Select
17 | 2006-07-21 | 2006-07-21 | Admin | Query | DefectTracking Default (| Default Query for table 'DefectTra | SELECT DefectTracking.*, | Select
18 | 2006-07-21 | 2006-07-21 | Admin | Query | SalesLeads Default Quer | Default Query for table 'SalesLea | SELECT SalesLeads.*, | Select
19 | 2006-07-21 | 2006-07-21 | Admin | Query | State Default Query | Default Query for table 'State' | SELECT State.*, | Select

You can create and delete queries and Reports directly in the grid – as you would do with any grid records. There are just a few restrictions on what you can do:
You cannot delete a Default query
You cannot delete a query that has Reports based on it
You cannot create a Report based on a Non-Select query

All columns in both the queries and the Reports grids are self-explanatory. The only values you can modify directly in the grids are the 3 last values for the Reports: Name, Caption, and Comment. To modify everything else, you must open the query or the Report in its own form. Double-click the corresponding record to do so. This will open the query in the Query form and the Report in the Report form.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

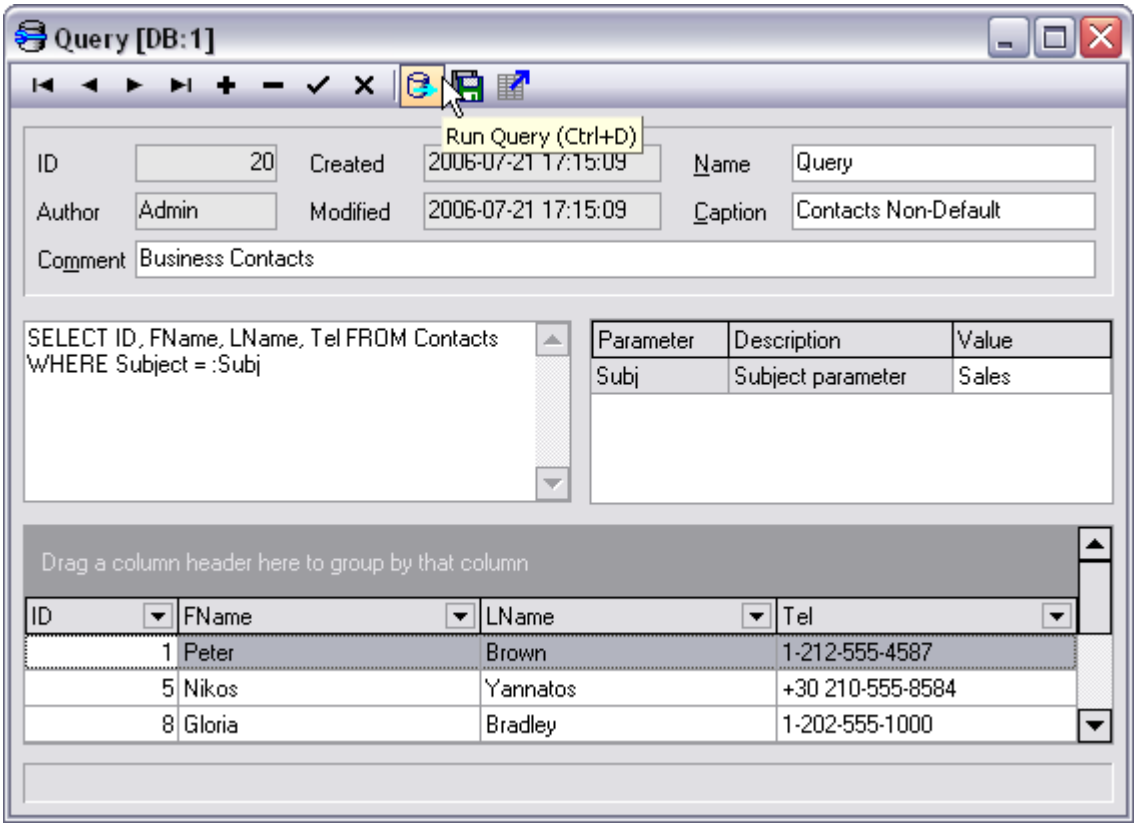
2.17 Query Form

We have briefly introduced queries in the preceding chapter on the Queries List. Here, we shall examine the topic of queries in more detail. This is an advanced subject, and if all you are interested in is producing a Report based on a query, you can skim through this material on your way to the next chapter that deals with Reports.

To recap, Alventis supports queries using a large subset of the ANSI SQL-92 language specification. Queries are per-Database, i.e., they have access to the tables of a single Database they belong to. Queries may be based on data tables of the Database. They may also "draw upon" some System tables, e.g., the table of Users. Queries are considered to fall into one of two categories: Select queries (the SELECT statement) and Non-Select queries (everything else, e.g., UPDATE, INSERT, and so on). Alventis automatically creates a set of Default queries for you: one per data table. Default queries may be used as source of Reports based on their respective tables. Default queries can be neither deleted, nor modified.

In the following discussion, we will be using selected SQL query examples, but we will not be explaining their syntax. You can consult the included SQL Reference if something is not clear. If you find the Reference too terse for your liking, you can also find some good books on the SQL language, most of which will apply to Alventis with few modifications.

The **Query form** is depicted below.



Copyright © 2002-2007 Alventis Corporation. All rights reserved.

At the top of it, we find the usual DataNav bar. On the right-hand side of the bar, there are 3 buttons that we have not seen before.

Run Query. Click this button to attempt to run the query or, said differently, execute its statement. Depending on the nature of the query and the amount of data it has to process, this may take a short or a long time, so you may have to be patient once you have clicked this button. Note that if the query is of the Non-Select type, it will proceed right away to modifying whatever data you instructed it to modify. There will be no "Are you sure?" prompts and no opportunity to cancel or undo your changes (short of restoring the original data from a backup), so proceed with caution.

Save As. Clicking this button immediately makes a copy of the current query. This makes it easy for you to make a clone of an existing query and apply whatever changes you want to the clone thus preserving the original query intact.

Export Results. This button launches the Data Export Wizard that allows you to export the currently-displayed results produced by your query to a delimited text file. We'll discuss Data Export in a separate chapter.

Just below the toolbar, we find a group of familiar Text Boxes with fairly self-descriptive names. As is usually the case, you can set Name, Caption, and Comment to whatever you find useful.

In the middle of the form we find the two main controls. The Query Box is a plain-text memo control. This is where you write the text of the query. SQL doesn't care about carriage returns, so you can format your query as you see fit.

Just to the right is the Parameters grid. Parameterized queries use parameters in their statements. Such parameters act a little like variables that allow you to leave the query statement unchanged but still modify the query's behavior by setting these parameters to different values. Parameters in SQL are identified by the colon ":" that precedes their names.

The query shown in the above screenshot has one parameter "Subj". Alventis places all parameters it finds in your query into the Parameters grid where you can assign the values you would like them to use when the query is executed. You can also write a little Description of each parameter. This is not used anywhere and is just for your convenience. Parameters are automatically created and deleted based on the query statement you write, you cannot create/delete/rename them in the Parameters grid.

Parameter data types are interpreted according to what the value looks like. Alventis tries to interpret the value as a number, as date/time, and if the above attempts fail, as a string. If you want to force a particular Parameter to be treated as a particular type, you can use the SQL CAST() function. For example, CAST(Xyz AS Integer). To force a value to be interpreted as string, you can simply enclose it in single quotes: '123' would then be seen as string (as opposed to 123 which would be seen as integer).

Parametric Queries are only supported for single Queries, not batch Queries consisting of multiple SQL statements separated by semicolons.

The bottom of the form is occupied by the Output grid. This is where a Select query displays the resulting records. Non-Select queries don't have any result in the form of records, so they will not display anything there. Once you have produced some Output results, you can modify the query if

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

necessary. For your convenience the old Output is not automatically cleared, so that you could refer to it while editing the query. It may no longer be up-to-date though, so Alventis displays a warning to this effect in the Status bar at the very bottom of the form.
The information shown in the Output grid is read-only.

You can usually open the data record displayed in the Output grid by double-clicking it. For this to work, Alventis must be able to figure out the ID value of the record. Simply put: the Output must include the ID field of the main table the query is written against.

If the database engine fails to interpret your query statement due a syntax error, the error will be displayed in the Status bar. If the error makes a specific reference to a particular position in the query statement, Alventis will place the cursor at the "offending" position so that you could see right away where things went wrong.

Saving a query (i.e., posting of its record) saves the query statement, the parameters, and their values. It does not however save the query Output results, if any.

2.18 Reports

Alventis supports printing via the use of **Reports**. In general, Reports provide a means of arranging and laying out collections of records, and sending thus merged result into the Report Output. The latter is nothing more than an Alventis Memo that allows you to print its contents or simply save it electronically for future reference. This is all very sweet and even true, but probably not very clear, so let us show you what we mean on a very simple example.

Let's say we have records that go something like this:

ID	FName	LName	Tel
25	John	Doe	123-555-6148
26	James	Smith	321-555-6644
27	Tom	Brown	147-555-7766

We will use a Report Template (we'll show you the real templates later, this is just an example) that could look like this:

User ID: [ID] Full Name: [FName] [LName] Tel: [Tel]

As you may have guessed, the above tells Alventis how to arrange the values of the records to produce the Report Output, which in our case would resemble:

User ID: 25 Full Name: John Doe Tel: 123-555-6148
User ID: 26 Full Name: James Smith Tel: 321-555-6644
User ID: 27 Full Name: Tom Brown Tel: 147-555-7766

Conceptually, what we've just demonstrated using this rather abstract example is precisely what goes on when you generate a Report. The essential steps one would go through to produce some useful Report Output are then:

Get the Data. That is, start out with the data that will go into the Report. This corresponds to the 3 records we had in the example above. We'll see where the real data will be coming from in a little while.

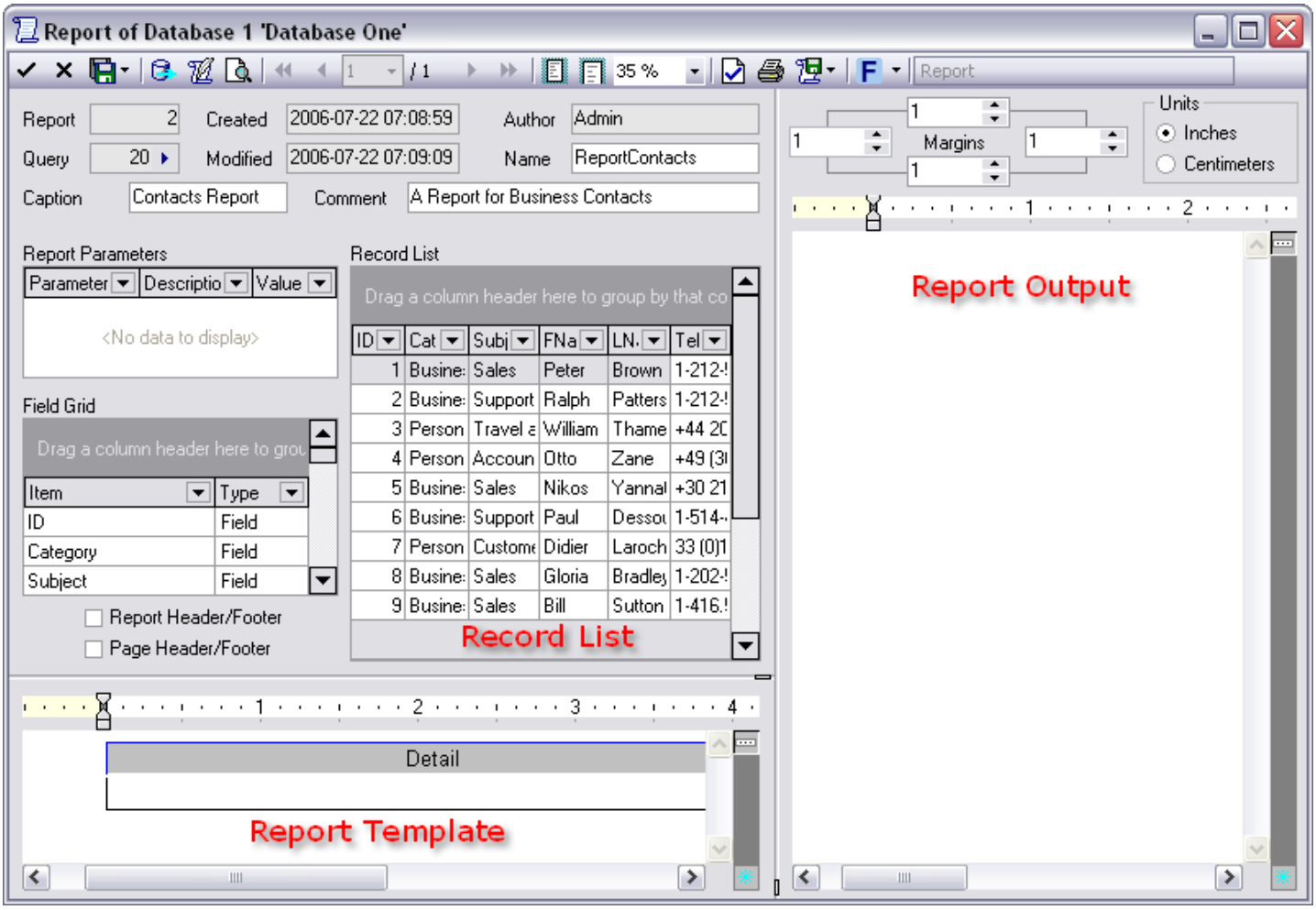
Arrange the Data. This is something we have not done in our simplistic example. We are talking about operations like sorting and grouping, but we'll examine them in more detail later.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Create the Template. This would roughly correspond to that single line "User ID: [ID]...". The real Report Templates are a bit more interesting than that.

Produce Report. That's just the name of the button you'd hit at this point.

Essentially, that's all there is to it! We will now examine each of the above steps in detail. To avoid too many "forward references" to material we haven't covered yet, we won't go over the Report form in a strict top-to-bottom order. Instead, we'll try to go through it in the most logical fashion: according to the steps in Report preparation that we have outlined above. To get at least some (relatively insignificant) part of the Report form out of our way, we'll just point out that the group of Text Boxes in the top/left area of the form should be mostly familiar to you by now, so we won't bother with them.



Getting the Data. As we have mentioned before, every Report is based on a Select query. The ID number of the query is listed in the Text Box called Query close to the top/left corner of the Report form. You can click the little arrow button in it to open the query in its own Query form. The role of the query is to produce an output in the shape of a collection of records. It is this collection of records that the Report will "use", i.e., take as its input, arrange the way you want, and eventually send to the Report Output. These records appear in the Record List grid (which in many ways resembles the Query Output grid).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

We have seen in the chapter on Queries that a query may have parameters. These parameters are listed in the Report Parameters grid. This little grid works exactly like its counterpart in the Query form. You assign the value you want to a parameter (if there are any) and update the Record List by re-running the query (the Re-run Query toolbar button accomplishes just that). This too is quite similar to what goes on in the Query form. The similarity goes even further: you can open a record in its own InfoView directly from the Record List if you like.

The Record List grid is obviously extremely important for the Report since this is what supplies the Report with the data. Its importance doesn't end there though. The Record List grid also represents the entire outline of the Report. This is what we were hinting at when we said earlier "Arrange the Data". Here's what this means:

Whatever records appear in the Record List grid will eventually end up in the Report Output. The implication of this is that no matter what cunning tricks you use to make certain records appear or disappear from the Record List, what ends up in the List is what ends up in the Output. The "tricks" you may use can be:

- Modifying the query itself that the Report is based on
- Adjusting the query parameters, if any
- Filtering the records directly in the Record List grid
- A combination of any of the above

You therefore have practically unlimited control over what data goes into the Report. The *what* is probably the most important issue and we have just established how you could control it, but there remains the *how* aspect of the Report.

How the Record List gets processed depends first and foremost on two aspects: Grouping and Sorting.

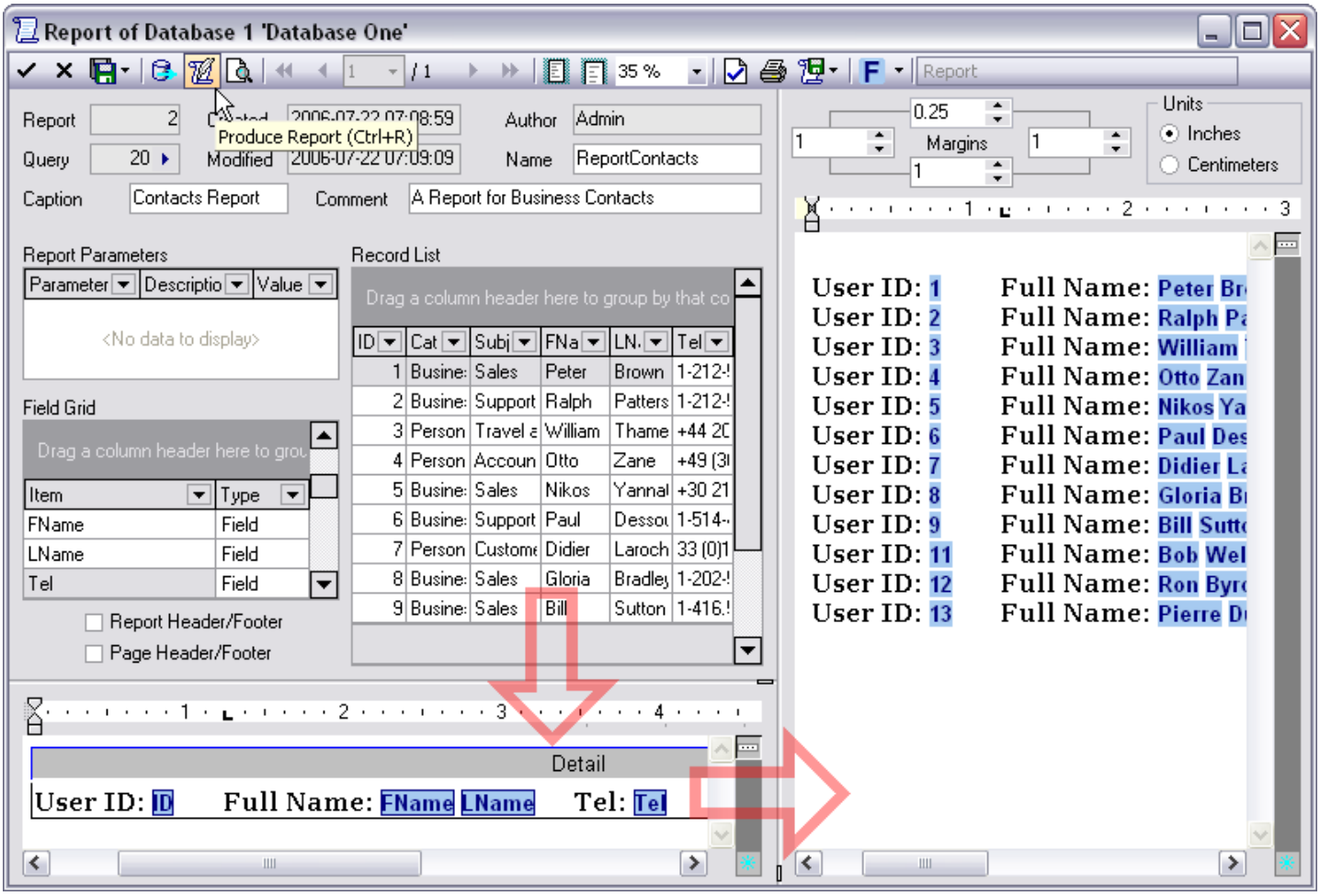
Grouping records in the Record List grid is performed exactly as you would do it in any grid in Alventis. The records in the grid become grouped by whatever column (i.e., field) you want. You can have as many grouping levels as you like, of course. The only thing that is special about the Record List grouping is that it directly affects the Report Template (that we will be examining shortly) and eventually the Report Output. As soon as you group by a certain field, the Template receives a pair of new Bands (or sections) that correspond to the grouping field: a Header and a Footer. We'll discuss Template Bands in more detail later. For now, it is sufficient to know that all grouping in the Record List grid directly "translates" into grouping of the Report Template and thus Output.

Sorting is similar: you perform it like you would for any grid, and most importantly, it has direct effect on how the records end up sorted in the Report.

To sum it up: take whatever records happen to be displayed in the Record List grid, and group and sort them any way you like. The results you see in the grid may not look anything like a Report but they exactly correspond to the structural outline the "real" Report will have.

Report Template. This is the focal point of the Report. You can think of it as a funnel through which we squeeze all the data that appears in the Record List grid, which squeezes out onto the canvas of the Report Output. Sounds fancy, but it's quite simple in reality.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



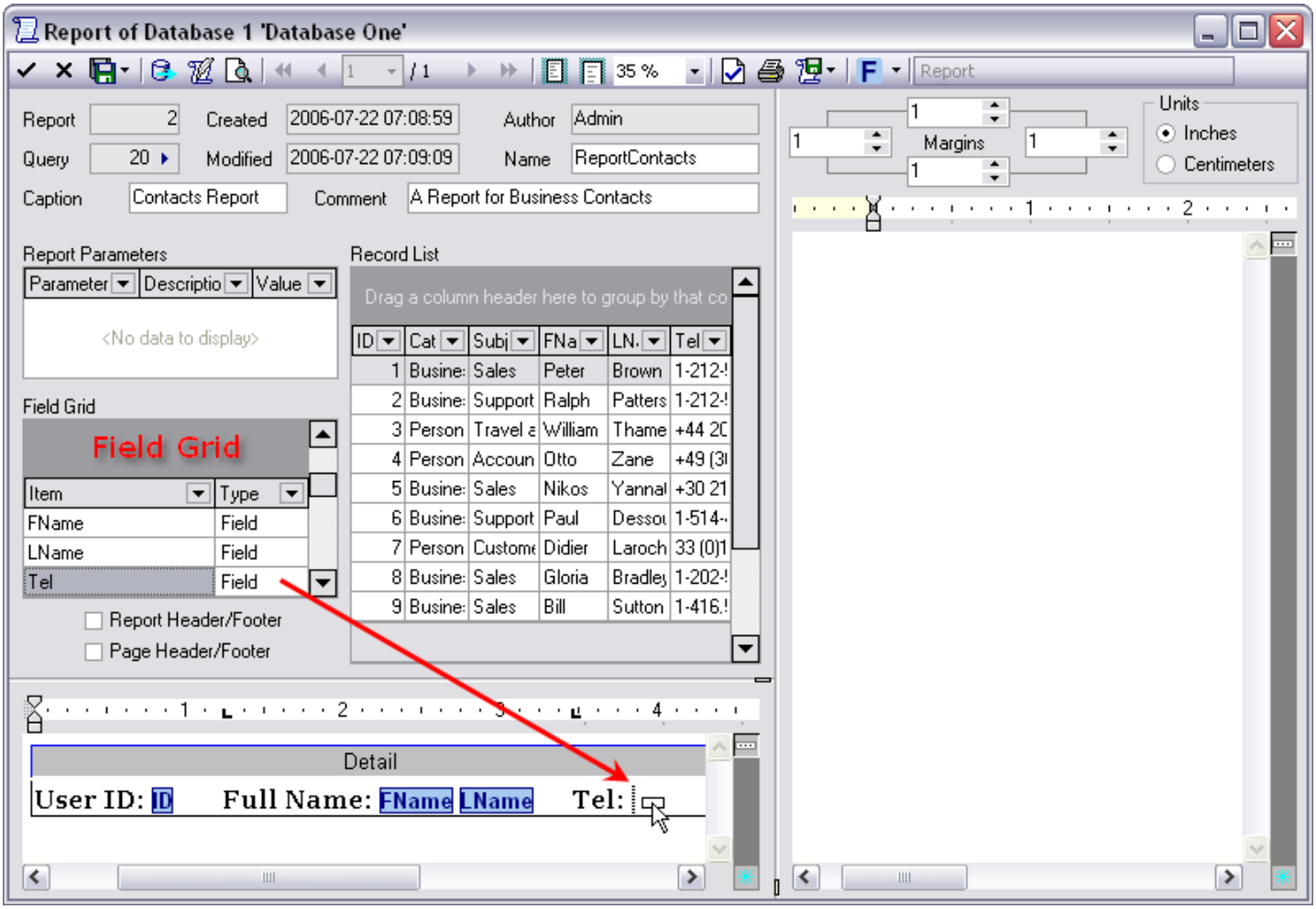
The Template is nothing more than a Memo which is divided into one or more Bands. Each Band looks and acts very much like a cell of a Memo Table in that it can hold whatever you want to place there: formatted text, bulleted/numbered lists, pictures, tables with more of the above, etc.

The simplest imaginable Report that would use no grouping would have the most minimalist Template with a single Band: Detail. The Detail Band corresponds to records in the Record List. Remember our abstract example where we used a line "User ID: [ID] Full Name: [FName] [LName] Tel: [Tel]" to produce some formatted output? This is precisely the kind of stuff we want in our Detail Band of the Template. As a matter of fact, this *is* what we want! Except that we can do better than write the field names by hand enclosing them in clumsy brackets. We can put the Fields themselves in the Template. This is what the Field Grid is for. It simply lists all Fields produced by the query – they also match the columns displayed by the Record List grid. The Field grid lists a few more things, but we'll get to them later. To add a Field to a Band of the Template, you simply drag-and-drop it from the Field grid into the Template – wherever you find suitable. To reproduce our example report, we would then:

- Type "User ID: " in the Detail Band
- Drag the ID Field from the Field grid and drop it right after the text we just entered.

The Field is created with a little border around it to help us distinguish Fields from other formatted text. Similarly, we could now add some more text and drag the remaining Fields (FName, LName, Tel – or whatever else we may want) where they belong.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Guess what: that's it! You have just created a real, albeit simple, Report. Well, at least the Template for one, but the really important stuff is all done. All that remains is to click the Produce Report button – and contemplate the results in the Report Output Memo (see the snapshot before the last one).

Report Grouping. We have already learned above that any grouping you perform in the Record List grid gets "applied" to the Report Template. Let's examine this in more detail.

To demonstrate grouping, we need some data we can sensibly group by some column/field. Let's just group our good old Contacts table by Category of contact: Family/Friend/Coworker. The data itself may look like this:

ID	FName	LName	Tel	Category
25	John	Doe	123-555-6148	Friend
26	James	Smith	321-555-6644	Family
27	Tom	Brown	147-555-7766	Friend
28	Jane	Smith	441-555-6144	Family
29	Bill	White	661-555-3316	Coworker

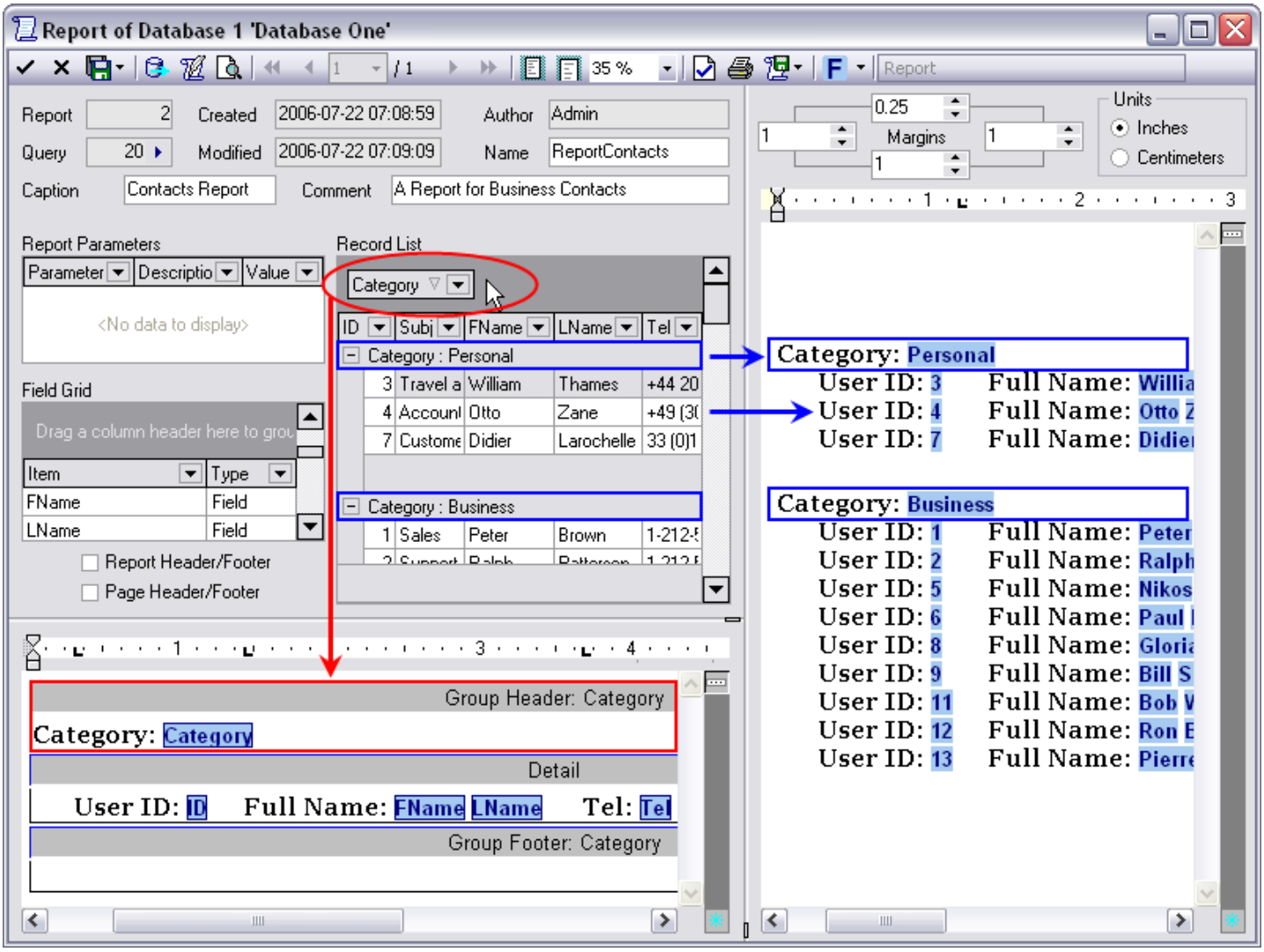
When grouped by Category, the above data will look like:

ID	FName	LName	Tel
Category: Family			

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

26	James	Smith	321-555-6644
28	Jane	Smith	441-555-6144
Category: Friend			
25	John	Doe	123-555-6148
27	Tom	Brown	147-555-7766
Category: Coworker			
29	Bill	White	661-555-3316

Something happened when we grouped the records in the Record List grid: two new Bands appeared in the Template: Group Header: Category and Group Footer: Category – one on each side of the Detail Band:



Let's arrange the new 3-Band Template in the following fashion and see what it produces. Let's put the Category Field in the Group Header: Category Band and leave the Group Footer: Category Band empty for the time being. Let's put the relevant record fields in the Detail Band. This would be our usual ID, FName, LName, Tel. Click Produce Report – and we obtain the output that essentially corresponds to the way the grouped Record List grid looked like. We have just effortlessly produced a Grouped Report. And how are *you* doing on your side?

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

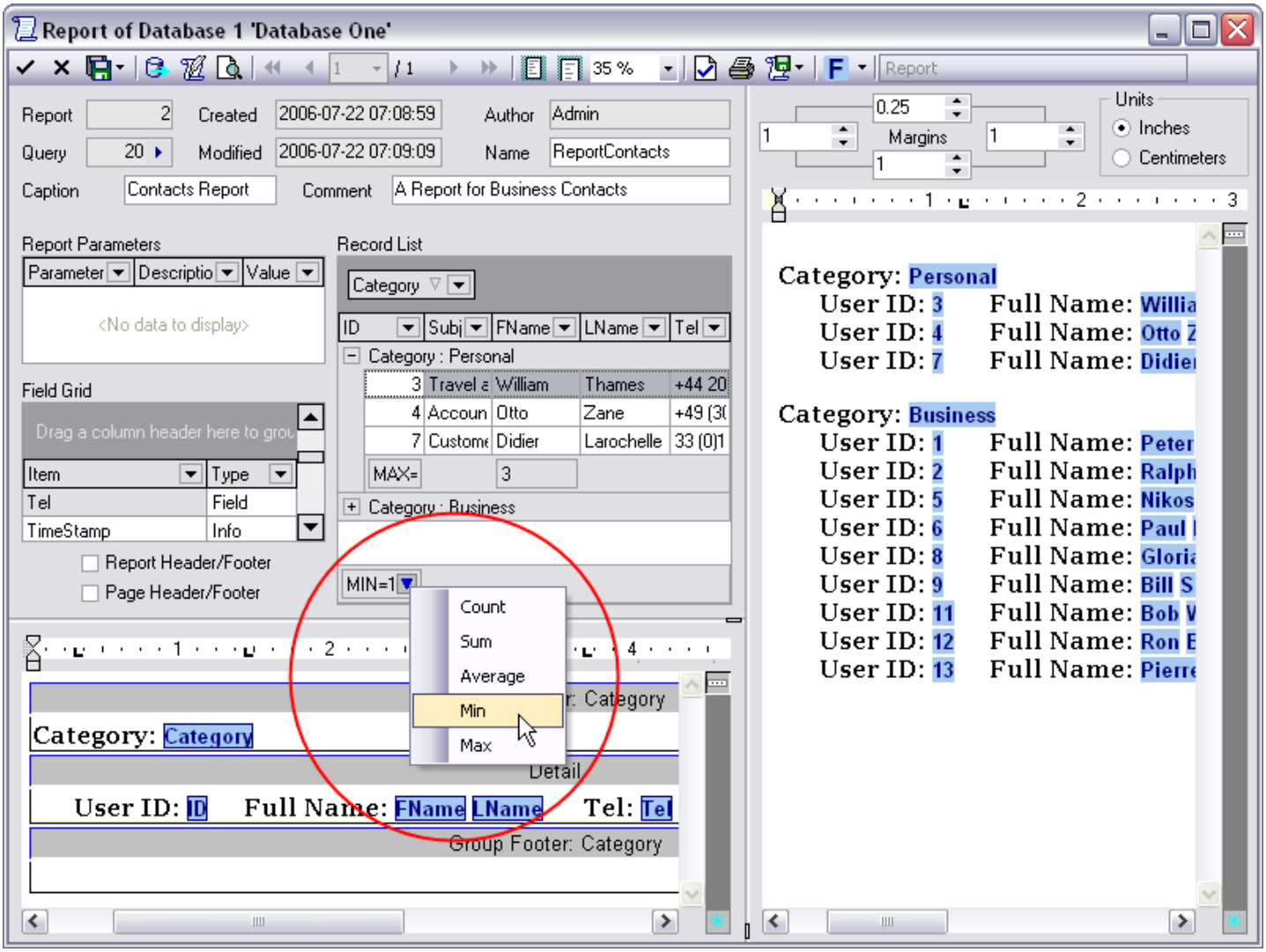
You can create as many grouping levels as you want. Simply group the data in the Record List grid, place the relevant Fields in the relevant Group Header/Footer Bands – and you're done. The "relevant" fields would typically be the ones that are responsible for the existence of the corresponding Band in the first place. Remember, in our example grouping by Category created the Group Header/Footer: Category Bands, and Category was precisely the Field we placed in the Header. This is most likely what you would want to do most of the time.

You can change the grouping order as you would do with any ordinary grid: just drag the columns around in the Group by Box. The corresponding Group Header/Footer Bands will be rearranged accordingly, preserving their contents.

Alright, so we have put something interesting in the Header Band, but what about the Footer? Is there anything useful we can place there? Turns out there is.

Summaries. As all grids in Alventis, the Record List grid supports dynamic creation of Summaries. It does so slightly differently from other grids, but the overall principle is the same. Here's how it works for the Record List grid.

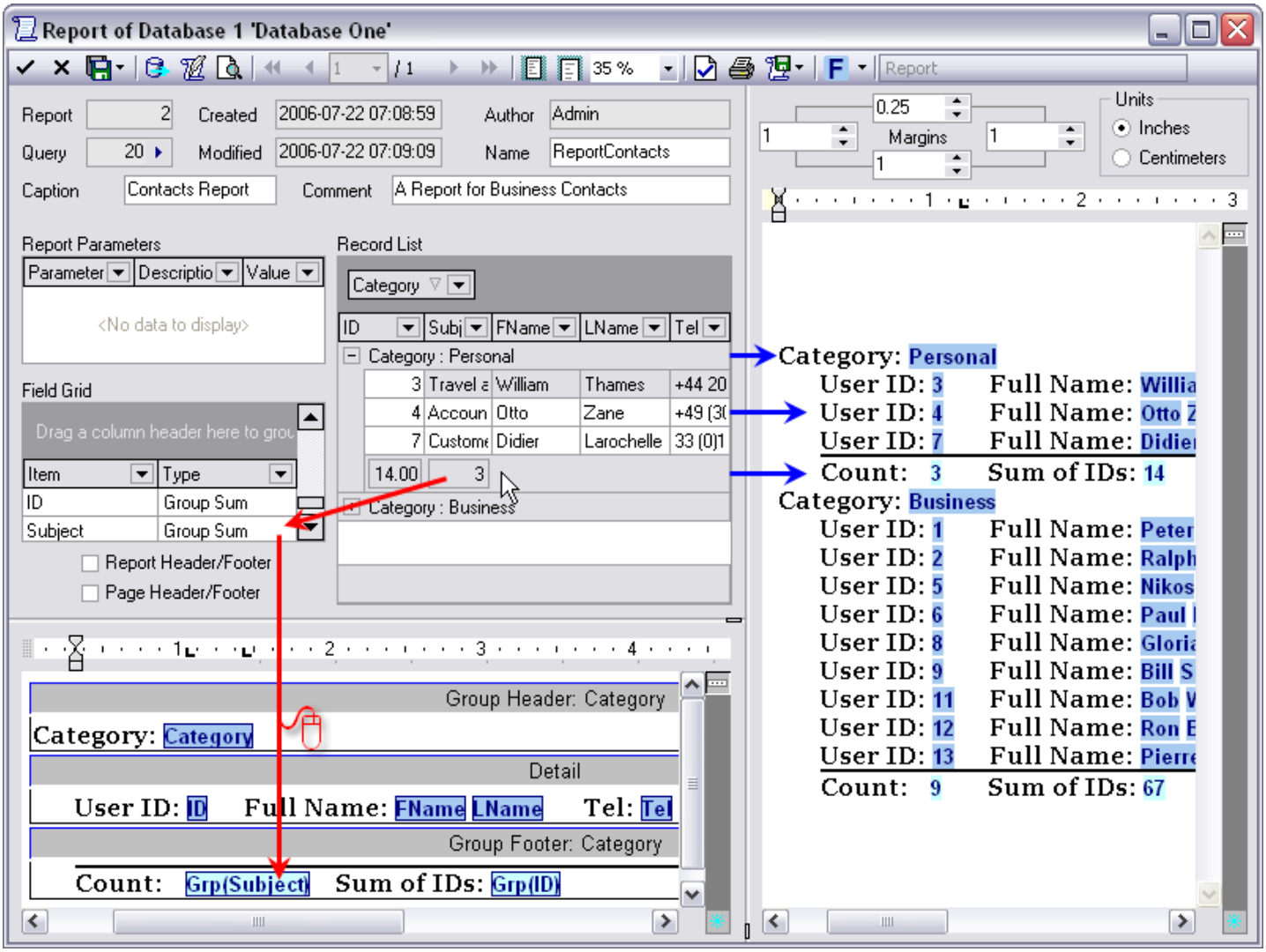
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Double-clicking on the Summary row toggles the Summary for the corresponding column On/Off.
Once a Summary is created, passing the mouse over it makes a little dropdown button appear close to the right edge of the Summary.
Clicking on that dropdown button displays a menu that allows you to directly select the mathematical operation you would like this Summary value to use.
As is generally the case, not all operations are supported for all field types (e.g., averaging string values doesn't make much sense), so see what is available on a case-by-case basis.
If a particular column only supports a single operation (e.g., strings only support counting of them), the dropdown button will be grayed-out.

You can create both Grid Footer Summaries and Grid Group Summaries.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Whatever Summaries you end up creating, they become listed in the Field grid – alongside with ordinary Fields themselves. You can drag-and-drop them exactly like any regular Field – wherever it makes sense in the Template. Normally, it would make sense in the Band that corresponds to the Record List grouping level that the Summary "belongs" to. In our earlier example, if after grouping contacts by Category we also created a Count (or any other) Summary counting contacts within each Category group, such a Summary would fit nicely in the Group Footer: Category Band. Or in the Header – if you prefer to precede a group of records with some cumulative information about them. Header and Footer Bands are basically "symmetrical" and in all respects interchangeable. You can spread whatever group-level information you want between these two Bands whichever way suits your taste, even repeat some or all of it in both if you like.

You may have noticed by now that there are a few elements in the Field grid that are neither Fields, nor Summaries. These are useful InfoFields:

TimeStamp is the time you hit Produce Report
Page is the current page number
Number of pages – we're not telling
User is the UserID of whoever is producing the Report

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Report ID/Name/Caption/Parameters/Filter will output the corresponding information pertaining to this Report. The first 3 are easy. The last two will list the Report Parameters that were in effect and the Record List Filter, if any. Including them in a Report may help the reader understand why the Report includes the records that it includes.

Since the InfoFields only appear in the Field grid, you can only drag them into the Template from there. Not so for Fields and Summaries. You can drag Summaries directly from the Record List grid. Fields are a bit trickier. Dragging a column caption sounds like a great idea, but this is something you do to group by columns. To distinguish an act of dragging a Field into the Template, you must Alt-drag-and-drop the column header. Hold down the Alt key and start to drag. You can now release the Alt key at any time. Drop the Field where you want it. You can obviously drag everything from the Field grid, so use whatever you find more direct and convenient.

Report/Page Header/Footer Bands. We have seen that grouping creates pairs of Bands: Group Header and Group Footer – a pair for each level of grouping. You may want to "surround" your output with just two more kinds of Bands.
Page Header/Footer Bands – as the name suggests – will appear at the top and bottom of every page (when printing).
Report Header/Footer Bands appear as "prologue" and "epilogue" of the Report.

You can enable each pair independently by checking the corresponding checkbox under the Field grid.
A Page Header/Footer could be just the right place for the Page and Number of pages InfoFields, as well as whatever else you may find useful there, like the Report Name.
The Report Header could be used as a title page.
The Report Footer may be suitable for the Report info, such as Parameters and Filter. But this is entirely up to you of course.

Certain items don't make sense in certain Bands though. For example, a real Field value has no place in any of the Bands we have discussed above. Such a value would not be known at the point in time when the Report Header/Footer or the Page Header/Footer Bands are produced, so Alventis won't let you place some items in Bands that are deemed inappropriate for them. Other than that, you are free to fully unleash your creativity.

Template Field Formatting. Since the Template is basically just a Memo, the full array of Alventis' formatting tools is available to you: Font, QuickStyle, Colors, and so on. You can lay things out using Memo Tables and/or Tab stops. Whatever works for you.

The only thing that is handled differently, are the Fields. You cannot edit a Field. You can delete a Field by selecting it completely and hitting Delete. You can move them around like you'd do with any text. And finally, you can format Fields. This works very much like formatting Hyperlinks: place the cursor anywhere within the Field (or select any portion thereof) and use any of the available formatting tools. The Field gets formatted in its entirety. Whatever text ends up replacing the little Field rectangle, it will be formatted with the Field's format in the Template. Aside from the little border surrounding the Field, that is.

You can select what format you want your Fields to have by default. The Report toolbar has a Font button dedicated to just this purpose. It is just like any other Font button in Alventis, except that whatever font it happens to have determines the font of all newly-created Fields. So, if you intend to format all or most of your Template's Fields the same way, set this Field Font button's font the way you want before you start dragging all those Fields into the Template – and they will come out the way you

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

wanted and won't require any "post-processing".
Field format is of course meaningless for Fields that represent (and thus will end up being replaced by) Memos and Pictures.

Memo Fields in Reports. There's nothing particularly special about them. They do have some features you should take into consideration though.
If all your records are known to have short Memos, you can do pretty much whatever you want. For example, arrange the record's Fields in such a way that several fields are in a table cell on the left, and the Memo is in another cell on the right – all within the Detail Band. Or maybe you have two short Memos that you would place side-by-side.
If at least some of your Memos are lengthy, you may want to avoid placing the Memo Field into a table's cell. As we have warned you elsewhere, Alventis can't correctly print a very long table cell that is longer than the page length. Such a cell will get cut-off at the bottom of the page – together with its contents. Your best bet in such a situation is to forego the use of tables to lay things out, and simply place a Memo Field directly in the Detail Band, most likely on a line by itself. This way, the Memo will not be constrained by a cell in the Report Output, so it will get properly paginated and will span as many pages as necessary.

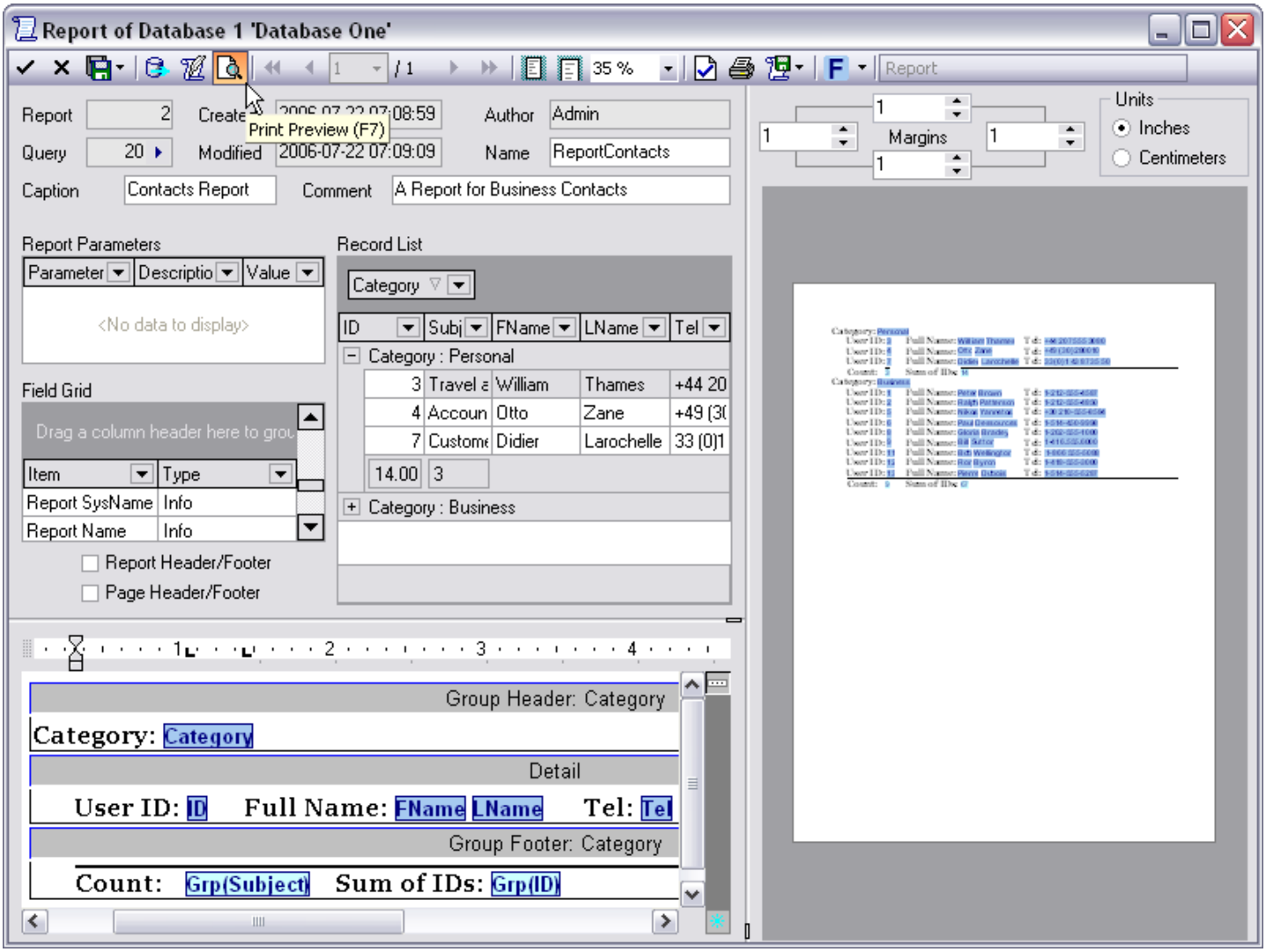
Picture Fields in Reports. When you place a Picture Field into the Template, what is placed there is a little Picture that simply states the name of its source Field. By default, when such a Field is replaced by the actual Picture from the data record, the Picture will be output with its "real" original size. This may or may not be what you want, and there are several things you can change. You can simply resize the little Field picture in the Template. The real Picture will be resized to fit whatever width you set for the Field. Only the width matters: the height is always ignored, preserving the original picture's aspect ratio. If your resized Field picture in the Template is wider than the real data Picture, the latter is not stretched though. If you do want it to stretch under such circumstances, you must explicitly set a special attribute of the Field picture. This attribute is accessible from the regular Picture Format dropdown dialog – but only for Template's pictures. It's called "Auto-Expand in Report" and you'll find it at the very bottom of the dialog. The Field picture's size must be anything other than default/100% for this attribute to be accessible. If this attribute is set, all data Pictures replacing this Field picture will be shrunk or stretched to end up with the same width.
Reset the Picture dimensions to default/100% to return to the default behavior, i.e., outputting Pictures with their original sizes.

You can also place the Picture Field in a table cell. The cell will constrain the output to its size. You probably wouldn't want to use a single-cell Memo Table for the sole purpose of constraining the Picture size, since you can achieve the same effect by simply resizing the Field picture. If the Field picture is already in a cell though, perhaps because you are using a table to better format the Report's layout, the cell-constraint feature comes quite handy.

Print Preview. Report Output is essentially just another Memo. You can even edit whatever the Report produced in there. All normal editing/formatting operations are available to you. Normally, you should not need to change anything there, but you can if you must. The Report Output Memo has just one little trick up its sleeve: this is the only Memo in Alventis that knows how to print its contents.

Printing per se works essentially the same way it works in just about every Windows application: you click the Print button, respond to your printer's dialog box if appropriate, and you're all set. But before you print anything, it's usually a good idea to preview the output, and that's what Print Preview is for.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Clicking the Print Preview toggle button on the toolbar places the Report Output Memo in Preview mode. You can't edit the Memo's contents in this mode. You can merely see the preview of what the printout with your current settings would look like – one page at a time.

In the Preview mode, you can navigate between the pages of the Output in several ways:
- You can use the PageUp/PageDown keys on the keyboard when the Preview is focused
- You can use the First/Prior/Next/Last toolbar buttons
- You can pick the page you want directly from a combo box
- You can type the page you want into the combo box

You can zoom in and out of the Preview using several methods:
- With the mouse, clicking anywhere in the Preview will zoom in on the clicked spot. Clicking again will zoom back out. Basically, the mouse clicks toggle the zoom level between the current one and 100%.
- You can click the Zoom to Full Page and Zoom to Page Width toolbar buttons that will adjust the magnification level accordingly.
- You can pick or type-in the zoom factor you want in the Zoom Percentage combo box.
- You can zoom in and out by clicking the Numeric +/- keys on the keyboard (the ones on the far right, in the Numeric keypad portion).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Note that it is only in the Print Preview Mode that you get to see the Page Header and Footer.
Note also that you can leave the Output Memo in the Preview Mode while you are producing Report Output, adjusting the Template, re-running the Report, and so on.

Page Setup. You may at some point want to change your page setup, for instance, change your selected paper size and/or Landscape vs. Portrait orientation. You can accomplish all this by clicking the Page Setup button and thus invoking the standard Page Setup dialog. Change whatever you see fit and press OK to see your new settings reflected in the Report Output.

Margins. The Report Output is also the only Memo in Alventis that has real meaningful margins. Since you are likely to find yourself adjusting them rather frequently, the controls that let you set them are located right above the Output Memo. You can also adjust the left and right margins using the Ruler. The Rulers above the Output Memo and the Template are synchronized, and you can use either one. All adjustments are "live", so you see the effect they have on your page layout immediately, in near-real-time.
While we're on the subject of margins, we'd like to point out something that is a little different about the Print Preview with regards to margins. In many word processors you may be familiar with, the Print Preview displays the whole white page on some shaded background, and the white rectangle you see corresponds to the entire sheet of paper sitting in your printer's input tray. In Alventis, the similar-looking white rectangle represents only the printable area of the page. Most printers cannot print on the entire page edge-to-edge. They usually have to leave small bands on some or all sides. Alventis simply does not include these "off-limits" bands, if any, in the white rectangle it shows you. The good thing about it is that you can know right away how much more usable space you have on each side: the entire white "margin" you see is usable, whereas with most word processors, only some part of it may be usable, and you don't have any indication, which part. The only drawback is that it is, well, different. Which is why we told you about it. That's all.

Page Breaks. Normally, your Report Output will get auto-paginated according to its contents and your page setup. If you are including any Memo Fields in the Report, and some of these Memos contain Hard Page Breaks, the Report will honor such page breaks – unless you place the Memo Field into a table cell (in the Template). The Hard Page Breaks, if any, will end up in a cell, and will therefore be ignored, since Alventis does not support page breaks inside table cells.
You can also insert Hard Page Breaks wherever you want in the Template. For example, you may want to separate the Report Header from the rest of it with a page break, so you'd simply put one at the end of the Report Header Band. Placing a page break at the end of a Group Footer would separate Groups.

You can save or cancel your changes to the Report Template and the Record List grid setup (grouping/sorting/summaries/filtering) by clicking the Post or Cancel toolbar buttons at the far left of the toolbar.

You can also create a clone of the current Report. Clicking the Save As button offers you two choices:
Duplicate Report under Original Query will create the copy of the Report attached to the same query as the current Report.
Duplicate Report under Duplicate Query will create a duplicate of both the query and the report.

Save Output. Clicking this button displays a list of all tables of the same Database where you could

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

"export" the currently-displayed Report Output. A table "qualifies" as a possible valid destination if it has at least one Memo field (and you have sufficient privileges to create records there). Whichever table you select from the list, a new record will be created in it, and the current contents of the Report Output Memo will be saved in that record's Memo field. This makes it very easy for you to save not only the Report's Setup (Template and Record List grid setup), but the Report Output itself. Maybe this will make printing the Report less necessary under some circumstances and will save a tree somewhere. If you find yourself using this feature regularly, you may want to dedicate a separate Alventis Table to Report Output – just to keep things better organized. Note that Page Header and Footer are not saved in this manner since you can only see them in the Print Preview mode.

In closing, you must have noticed that the Report form is divided into 3 panes with RotoSplitters between them. Each of the panes is a zoomable Panel, so you can double-click any of them to temporarily maximize them within the Report form.

2.19 Search Syntax

UniGrid Search supports powerful searches using a variety of expressions. We will give you an overview of their syntax in this topic. We are basing it on the documentation written by Tamarack Associates for their Rubicon component which Alventis uses for its full-text indexing needs.

Searches support the following operators (listed in the highest to lowest order of precedence):

LIKE, NEAR
NOT
AND, OR

Note that we are writing the operators in uppercase simply to make them stand-out from surrounding text, if any. The entire search text is case-insensitive, which applies to both literal text you are searching for and operators, so in "real life", a search for "Jack AND Smith" is equivalent to "jack and smith".

The following expressions show the syntax of each operator:

LIKE <string>
<string> NEAR <string>
NOT <expression>
<expression> OR <expression>
<expression> AND <expression>

In the above:
<string> is a literal string of text you are looking for, possibly ending with a wildcard character (e.g., maple, map*)
<expression> can be a <string> or another, simpler, expression, possibly enclosed in parentheses

This should become clear from some examples:

Search	What it looks for
maple	The literal word (string) "maple"
map*	All words that start with "map": map, maps, maple, etc.
ma*e	Words that start with "ma" and end with "e": maple, marble, etc.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

ma?le	Words that contain any character in place of the "?" symbol: maple or matle, but not marble
LIKE maple	Words that sound like maple, e.g., maple, maphle (whatever that is). Like compares words based on their phonetic similarity using a modified Soundex algorithm.
maple AND leaf	Text that contains both words anywhere
maple AND NOT syrup	Text with "maple" but not "syrup"
maple NEAR leaf OR flag	Text where either "maple" is found close enough to "leaf", or "flag" is found somewhere
maple NEAR (leaf OR syrup)	Text where "maple" is found not far from either "leaf" or "syrup"
"maple leaf"	The literal phrase – just as specified, except for quotes (i.e., it will find: a maple leaf fell from a pine tree).
(ma?l* AND leaf*) OR LIKE flagh	You should be able to figure this one out by now, but if not: text that contains either some word that resembles "flagh" (e.g., "flag") or, failing that, text that contains both some word whose first two letters are "ma" followed by any one letter, followed by "l", followed by whatever -and- any word that starts with "leaf".

As you may have gathered, you can use two wildcard characters:
* means "any sequence of characters" (or none)
? means "any single character"
You can freely mix-and-match them as you wish, e.g., *uto??b* would match "automobile" or "utofobia" (if such a thing existed).

You can enclose phrases or words you want to find in a particular sequence in double or single quotes: "good dog" is equivalent to 'good dog'. Whichever quotes you use, you must always make sure you match them properly, i.e., that you have proper pairs of quotes of the same kind. Same goes for parenthesis.

There's also one simplified type of search you can perform. If you don't need to use any expression operators, you can simply make a search consisting of a space-separated list of words, and the search will use the AND logic for them. For example, the following two searches are equivalent:

maple syrup pine
maple AND syrup AND pine

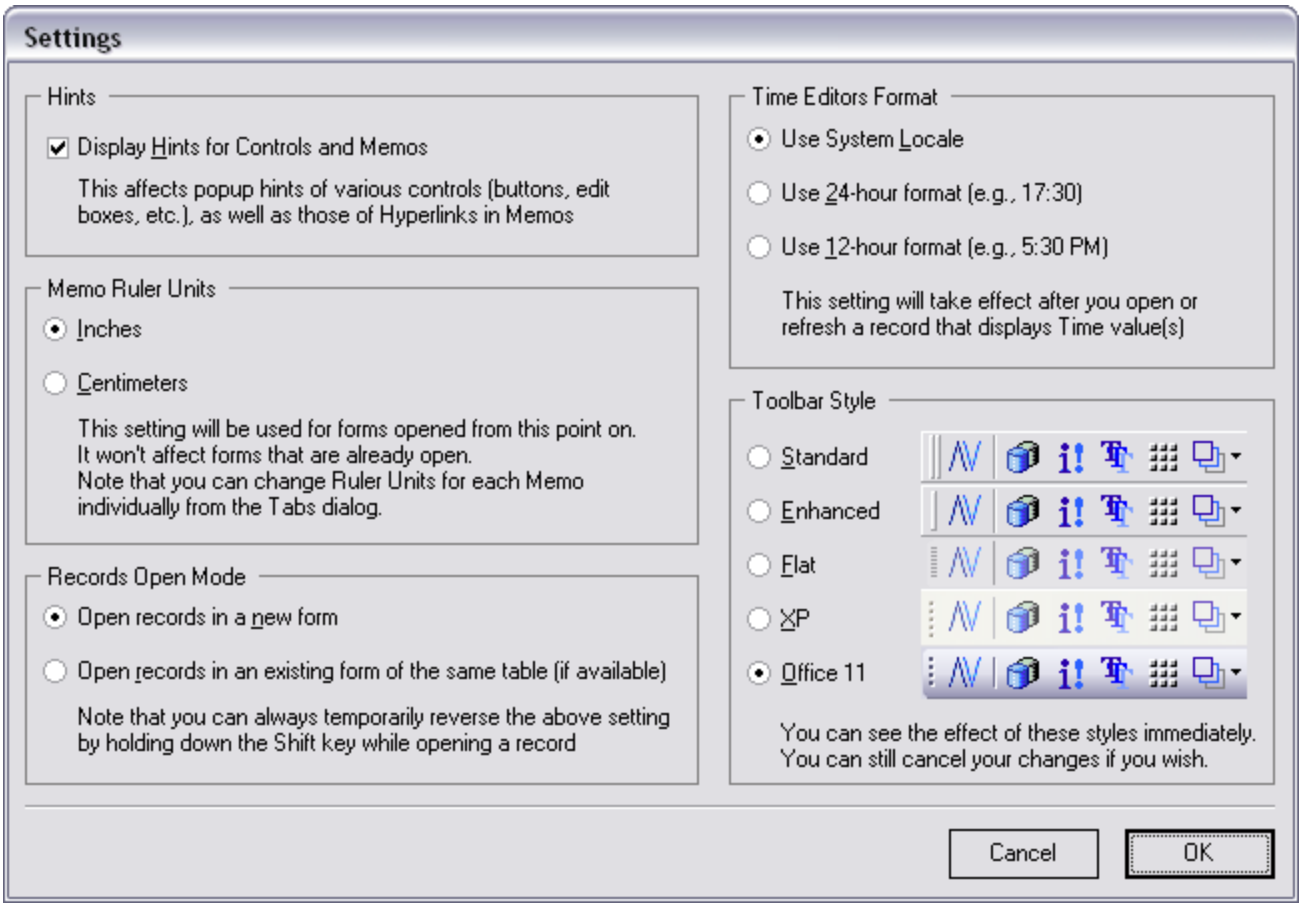
If you write a search that Alventis cannot understand according to the syntax described above, the search won't be executed, and Alventis will make the search text red as an indication of an error condition. This may happen periodically while you are in the process of typing your search. For instance, if the search you intend to execute is "maple AND syrup" (with no quotes), the moment you type "maple AND", the search expression becomes red indicating an error. This makes sense since the above expression is indeed syntactically incorrect. But since you know you'll "fix" it in a moment by adding "syrup" to it, you should simply ignore such transient errors.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

There are two search types that are more time-consuming than the rest: searches where the expression starts with a wildcard (e.g., "*ortoise") and proximity searches (e.g., "slow NEAR search"). They may take on the order of 1 second per 1000 records, so it only becomes a problem for large tables.

2.20 Alventis Application Settings

Invoke the **Settings** command under the File menu to open the Settings dialog. The settings are few and quite self-explanatory, so we will quickly go over them.


Settings
Hints
Display Hints for Controls and Memos
This affects popup hints of various controls (buttons, edit boxes, etc.), as well as those of Hyperlinks in Memos
Memo Ruler Units
Inches
Centimeters
This setting will be used for forms opened from this point on.
It won't affect forms that are already open.
Note that you can change Ruler Units for each Memo individually from the Tabs dialog.
Records Open Mode
Open records in a new form
Open records in an existing form of the same table (if available)
Note that you can always temporarily reverse the above setting by holding down the Shift key while opening a record
Time Editors Format
Use System Locale
Use 24-hour format (e.g., 17:30)
Use 12-hour format (e.g., 5:30 PM)
This setting will take effect after you open or refresh a record that displays Time value(s)
Toolbar Style
Standard
Enhanced
Flat
XP
Office 11
You can see the effect of these styles immediately.
You can still cancel your changes if you wish.
Cancel
OK

The Hints checkbox controls the display of ToolTips when you pause the mouse over various user interface items. It is equally responsible for the ToolTips for Memo Hyperlinks, so you may want to leave it enabled.

Memo Ruler Units offers you a choice of inches or centimeters. It will be used as default unit for newly-opened Memos.

Records Open Mode has been described in the UniGrids chapter.

Time Editors Format offers you 3 options that affect how time values are displayed in Time Edit Boxes. Use System Locale is the default option which will use your Windows system settings as specified in the

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Time tab of the Regional Options Control Panel applet.

Toolbar Style offers a choice of 5 visual styles that affect the appearance of all toolbars and menus in Alventis.

2.21 New/Open Record Wizards

You can create new records or open existing ones using the following commands:

New Record Wizard

Open Record Wizard

The Wizards are quite simple and intuitive, so you should have little trouble finding your way around them.

It should be noted that there are other, often more efficient ways to accomplish the same tasks.

You can create new records:

- By clicking the button in the DataNav toolbar of an InfoView
- By double-clicking the desired table in the Tables grid of a UniGrid

You can open an existing record:

- By double-clicking it in the Search Results grid of a UniGrid
- By focusing it in the Search Results grid and hitting Ctrl-Enter
- By dragging it from the Search Results grid onto an already open InfoView

All of this is described in more detail in the topic on UniGrids.

2.22 Import/Export

2.22.1 InfoSet Import/Export

Alventis offers you a facility to easily exchange data in Alventis' proprietary format with other Alventis users. You can also use this same functionality as means of creating selective data backups or "snapshots".

An InfoSet is a data table plus the InfoView forms that are based on it. Very simple. For the Contacts table, for example, this would be the Contacts table itself + whatever InfoView(s) may be "attached" to it.

To save and/or share something you have in Alventis with the rest of the world (or with a backup diskette, or whatever the reason may be), you can export any of the following:

- The data table per se. This is a minimum. You can export the table with or without data, i.e., just the table structure (what fields it has, what their properties are, etc.) or the actual data as well.
- The accompanying InfoView forms, if any. This is optional.

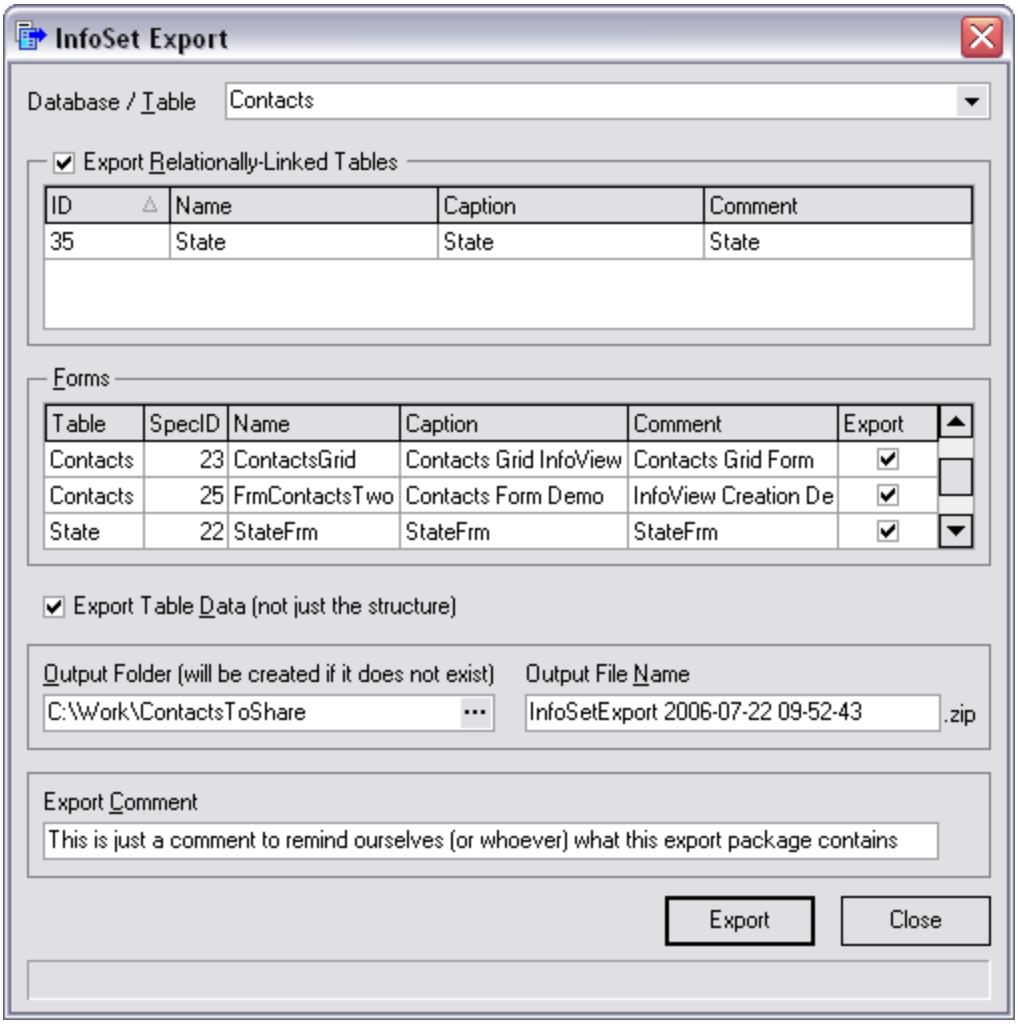
Whatever you end up exporting, we will refer to it as InfoSet (even if there are no InfoViews that come

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

with it).

The basic concept is quite simple. Exporting an InfoSet saves a group of files relevant to what you choose to export. Importing an InfoSet imports it from such a group of files into some Alventis Database. Let's start by examining InfoSet Export.

InfoSet Export is performed by clicking the Export InfoSet button. This launches the InfoSet Export Wizard, which is so magical it only has a single screen depicted below:



At the very top is the Database/Table combo box where you select what table from what Database you want to export.

Some Alventis tables may be relationally linked. We discuss Relationality elsewhere. All that's important to understand is that if Alventis notices that there are any tables that the selected table references, such tables will be listed in the grid.

The Export Relationally-Linked Tables checkbox, when checked, will export not only the main table you selected, but also all the tables it references. If you know what you are doing and understand the

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

implications, you can uncheck this checkbox, thus only exporting the main table.

The Forms grid lists all InfoView forms that "belong" to all the tables you are exporting – the main table and the relationally-linked ones, if any. You can export whatever forms you want by checking/unchecking the Export checkbox for each form.

The Export Table Data checkbox, when checked, will make the Wizard export the data of all exported tables.

Output Folder should be set to the path where you want to export the InfoSet. It is preferable that you set it to either an existing empty directory or specify a new directory. Alventis will attempt to create it in the latter case.

Output File Name is the name you want for the export file. Whatever gets exported (and it can be quite a few files) will be zipped by Alventis in a standard Zip file with whatever name you specify here. You don't need to add the .zip extension: Alventis will do it for you. If you end up specifying a file name that cannot be used (e.g., because you include some illegal characters in it), Alventis will use the default file name which is simply "InfoSetExport" followed by the date/time of the operation.

Export Comment gives you an opportunity to attach a comment to the exported InfoSet. Whatever may be useful when you or somebody else decides to import it.

As soon as you specify a proper Output Folder, the Export button becomes enabled, and clicking it will perform the requested operation. Essentially, that's all there is to it.

Multi-InfoSet Export. You can export a single InfoSet as we have seen above. You can export more than one InfoSet following the same exact procedure. If you export multiple InfoSets in the same Output Folder, this will simply create a Multi-InfoSet Export package, which will still end up as a single Zip file. You can of course export one InfoSet to one Output Folder, and another – to some other directory, if that's what you prefer.

Multi-InfoSet Export must be performed and completed without closing the InfoSet Export Wizard though. The reason for this is that whatever you export during a Wizard's "session" is zipped according to the settings you have specified at the time when you close the Wizard. Once the output has been zipped, you can't add anything to it, so subsequent Export operations – even into the same Output Folder – will be treated as separate and will result in new, separate Zip file(s).

A word of caution. You should never combine multiple InfoSets from more than one Database in a single Export package. It might work or at least seem like it has worked fine (if all relevant table and InfoView IDs happen to be unique), but generally-speaking this is an unsupported operation that you should not attempt.

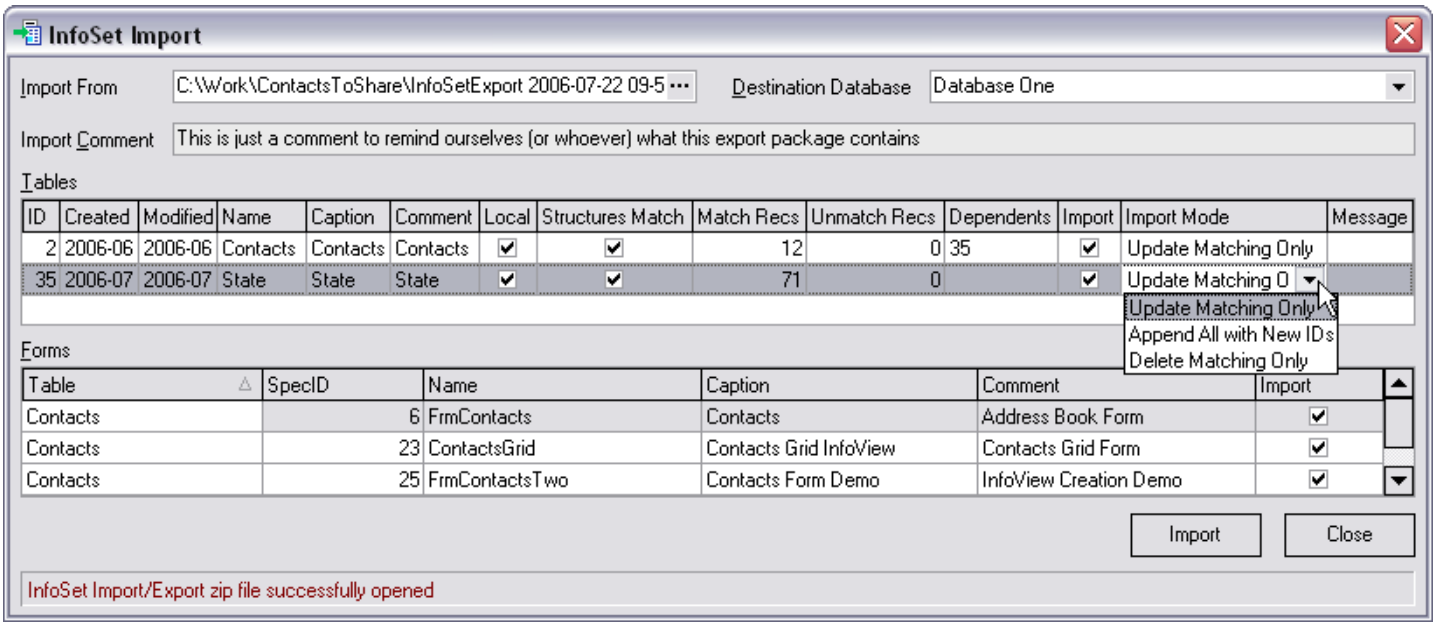
Technical note. Alventis may create and leave a hidden file called dbisam.lck in the Output Folder. You may not be able to delete this file until you exit Alventis. You can delete it manually once Alventis has released its "hold" on that file.

It is now time to see how we can import back into Alventis something previously exported.

InfoSet Import is performed by clicking the Import InfoSet button. This launches the InfoSet

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Import Wizard. In the spirit of Alventis, this Wizard too accomplishes all of its sorcery with a single screen depicted below:


InfoSet Import
Import From
C:\Work\ContactsToShare\InfoSetExport 2006-07-22 09-5
Destination Database
Database One
Import Comment
This is just a comment to remind ourselves (or whoever) what this export package contains
Tables
ID Created Modified Name Caption Comment Local Structures Match Match Recs Unmatch Recs Dependents Import Import Mode Message
2 2006-06 2006-06 Contacts Contacts Contacts 12 0 35 Update Matching Only
35 2006-07 2006-07 State State State 71 0 Update Matching O
Update Matching Only
Append All with New IDs
Delete Matching Only
Forms
Table SpecID Name Caption Comment Import
Contacts 6 FrmContacts Contacts Address Book Form
Contacts 23 ContactsGrid Contacts Grid InfoView Contacts Grid Form
Contacts 25 FrmContactsTwo Contacts Form Demo InfoView Creation Demo
Import
Close
InfoSet Import/Export zip file successfully opened

Before we go any further, you should remember that an Import operation will modify your local data. If you ever make a mistake, such modification can lead to drastic results, including data loss. You should therefore have a secure recent backup of your data before you attempt importing something into it. This said, let's proceed.

Destination Database is the Database into which you would like to import something. It may not be the first control on the form in the left-to-right order, but you may still want to pick the Database first. Nothing terrible will happen if you don't. Except that under some circumstances, the Wizard's initial inclination to import (or not) a particular table may be influenced by the currently selected Destination Database, so you might have to do more work later (check a checkbox it decided not to check for you, that is).

Import From is a Text Box where you specify the name of the Zip file that contains the InfoSet package you or somebody else has at some point in time exported from Alventis. You must specify the full filename with the path. You can use the little dotted button to browse for the file you want using the standard Open dialog.

Import Comment is a read-only box that displays the Comment that was specified by whoever prepared the InfoSet package.

As soon as you enter a proper Zip file name in the Import From box, the rest of the form comes to life according to what InfoSet(s) the specified Zip file contains and what your chosen Destination Directory is.

The Tables grid displays a list of all InfoSet tables that were found in the package. We'll go over its columns now.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

ID is simply the table's ID – whatever it happened to be in the Database from which someone originally exported it.

Created/Modified/Name/Caption/Comment are the usual information-only columns.

Local indicates if you have a table with the same name locally, i.e., in the selected Destination Database. InfoSet Import cannot rename tables. If you are importing a table called Contacts into a Database that already has a table called Contacts, this in itself is fine, but Alventis understands your intention as one to transfer some data from the Contacts table in the package into your local Contacts table. This is precisely what the Local checkbox indicates.

Structures Match. If Local is true, i.e., you have a local table with the same name, the structures of the table you are importing data from and your local table must match exactly, as if this were the same table. The Structures Match indicates if this is indeed the case. It is only applicable when you do have a local same-name table, so it will be grayed-out if you don't.

Match Recs indicates the number of records with matching IDs. This is of course only applicable if you have a Local same-name table.

Unmatch Recs indicates the number of "incoming" records with IDs that you don't have locally. This value is meaningful regardless of whether you have a Local same-name table. If you don't, it will simply equal the total number of "incoming" records.

Dependents will list each table's dependent or linked tables, i.e., other tables it may be referencing, if any. When importing Relational InfoSets, you must import a table together with whatever tables it references. This column will give you an indication if this is the case.

Import checkbox is finally something you have direct control over: check it to enable the import from the corresponding table. When importing from Multi-InfoSet packages, you may want to perform the import from only some tables, so this is where you'd specify which ones.

Import Mode deserves special attention, so we will explain it shortly, once we've covered the rest of the form.

Message will display hopefully helpful hints or explanations the Wizard may have regarding each table. For example, if a particular table cannot be imported (e.g., due to a structure mismatch) this is where it will tell you.

The Forms grid lists all associated InfoView forms and is luckily quite simple.

- Table tells you which table the InfoView "belongs" to.
- SpecID is the original ID number of the InfoView at the time it was exported.
- Name/Caption/Comment are what they seem.
- Import checkbox allows you to tell the Wizard to import some but not other forms.

Import Modes. Depending on various factors, certain import operations may or may not make sense or even be possible. These factors include: Local, Structures Match, Match Recs, Unmatch Recs. Together, they tell Alventis what can and cannot be done with a particular table or its data. The specific logic used is quite complex but it simply follows the "common sense" logic of the situation, so you can usually figure out fairly easily why only certain choices are presented for certain tables – if you feel like it of course.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The possible values or choices you may see are:

- Can't Import. This simply indicates that importing is impossible for some reason.
- Nothing to do. This means the Wizard arrived to the conclusion that the table in question doesn't need any import (likely because there are no records to import and the structures already match).
- Structure Only. This choice allows you to import the table's structure but not the data.
- Append All / Preserve IDs. All records will be appended with their ID values preserved.
- Append All with New IDs. All records will be appended but their ID values will be auto-set to whatever the next "empty slots" in the local table happen to be.
- Skip Matching, Append New Ones Preserving IDs. Almost entirely self-descriptive, this means that any matching records with same ID values will be ignored completely, and any new records for which there is no local matching ID will be appended with their "incoming" IDs preserved.
- Skip Matching, Append New Ones Resetting IDs. As above, but the "incoming" IDs will get auto-set.
- Update Matching Only. Records with matching IDs will be updated (local ones will be overwritten with whatever is in the "incoming" ones), nothing else will be done with unmatching records, if any.
- Update Matching, Append New Ones Preserving IDs. As above, but records with IDs that don't exist locally will be appended preserving their ID values.
- Delete Matching Only. This seldom-used option will delete whatever local records have a matching "counterpart" with the same ID in the import package. Nothing gets imported.

If you just read the above list of Modes for the first time and don't have much experience importing InfoSets, it may look quite daunting. In most cases though, you won't have to face too many complicated choices, and never all of them at the same time for the same table. Alventis always tries to suggest the most common Mode. In some cases more than one may indeed be available, and it is only you who can decide what makes most sense based on your knowledge of what you are importing where. We may not be able to make that choice for you, nor even provide you with an exhaustive list of possible scenarios and suitable Modes, so you may occasionally have to resort to common sense and decide what you want to happen to matching and unmatching records.

The most basic reasoning is therefore this. Some incoming records may be "matching", i.e., their IDs match those of the locally-available ones. Others may be "unmatching" with no local counterparts. The number of such records will be displayed in the Match Recs and Unmatch Recs columns respectively.

Matching records can be skipped, imported "over" the local ones (update), imported with their IDs changed (in case, for example, the IDs match by accident and don't indicate any true relation between local and incoming records), and lastly you can just have the local ones deleted. There's no other conceivable operation you can perform on the matching ones.

Unmatching records may be skipped, imported with their IDs preserved (since they don't have any local matches, so the local ID values are "available"), or imported with their IDs changed.

Most sensible combinations of the above are available, so you can usually pick the right Mode based on how you want to handle the matching and unmatching records.

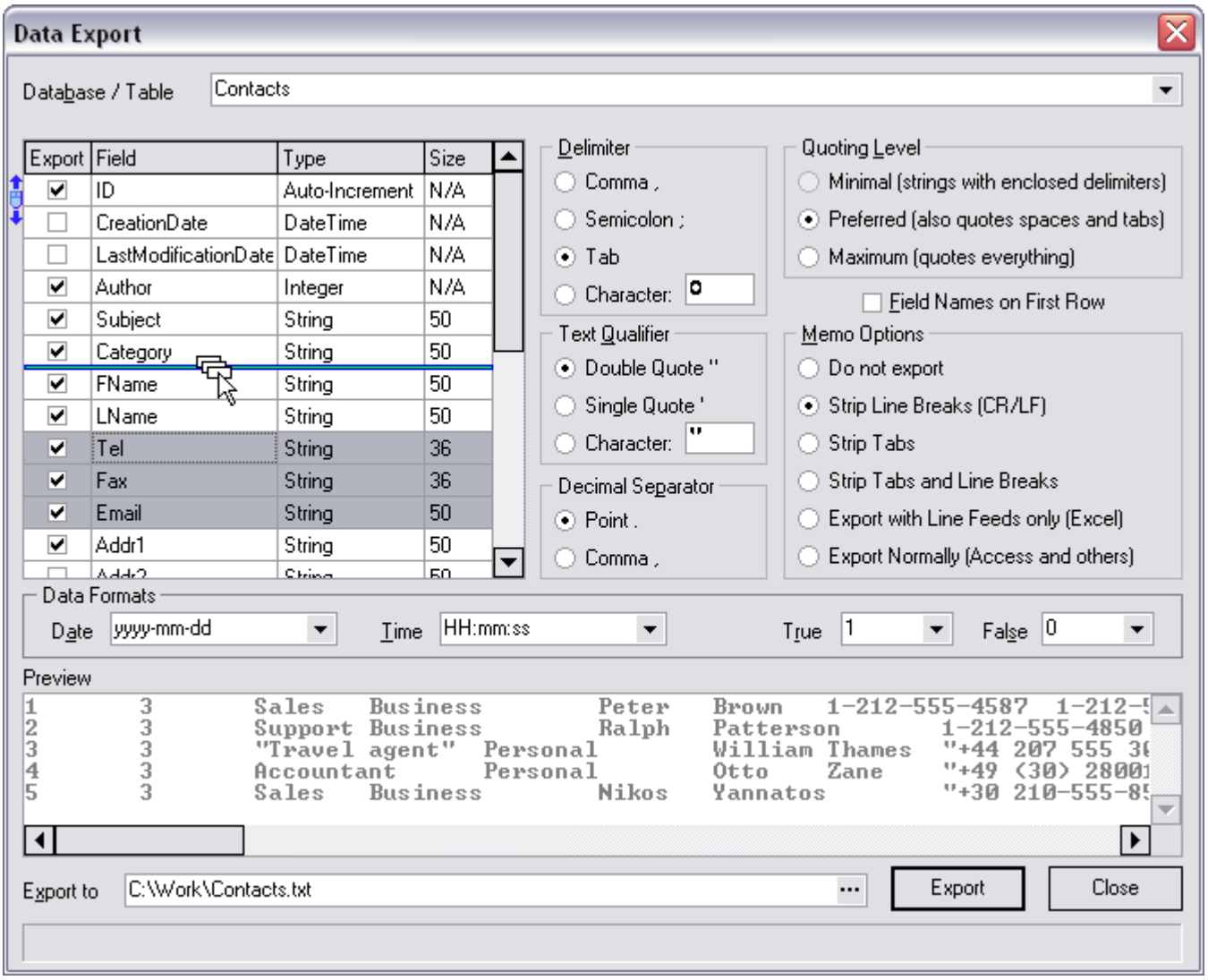
You should also remember that ID values may have a special meaning for some records. This is especially true of relational tables. The table in which some other table performs "lookup" using the record ID values must obviously preserve its IDs, otherwise relationships between data records risk getting broken.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

2.22.2 Data Import/Export

In the previous chapter on InfoSet Import/Export we've seen that you can export and import entire Alventis tables with their accompanying InfoView forms. This is the best way to share information between multiple copies of Alventis, but it won't be of much help if you want Alventis to share data with any other application: to share something they need to "speak the same language", which in this particular case means sharing a common data format. Data Export gives Alventis the ability to export its data in one of the most standard data exchange formats, and Data Import allows it to read several popular formats of data that may be coming from other applications.

Data Export. Clicking the Data Export button launches the Data Export single-screen Wizard. It allows you to export a table of your choice into a delimited text file.



The Database/Table combo at the very top lets you select the table whose data you would like to export.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

As soon as you make a selection, the rest of the form is automatically filled-out according to the contents of that table.

The Export Field grid on the left lists all fields of the table.
- The Export checkbox lets you specify which fields' values you want exported. You may uncheck any ones you don't need.
- Field is the name of the field.
- Type and Size are the field's data type and size respectively.

The exported values will be written to the output file in the order in which the fields are listed in the Export Field grid. You can adjust this order very easily by dragging fields up and down in the grid with the mouse (even several at a time).

The Delimiter group to the right of the grid lets you select what delimiter you would like to use for the destination file. The most common ones (Comma, Semicolon, Tab) are available to you by simply selecting the appropriate radio button. You can use some other delimiter of your choice by specifying it in the Character Text Box. You can input either a literal character or its numerical ASCII code, which you can enter in decimal or hexadecimal format (0x2F or $2F, for example).

Text Qualifier is the Quote character you'd like to enclose values in, when necessary. Normally, this would be a double or single quote character, but you can choose whatever you want in the same way it has been done for the Delimiter Character above.

Decimal Separator will be used in exported numerical values and you have a choice between the Point and the Comma.

Quoting Level specifies how "aggressively" you want your output values quoted. The options are self-explanatory.

Field Names on First Row checkbox allows you to include the field names as the first row of the exported text file.

Memo Options control how Memo text will be exported, if at all. Alventis Memos can obviously contain many elements that cannot be exported to a plain-text file, e.g., formatting, pictures. Alventis can only export the textual contents of the Memos. Many applications cannot import multi-line Memos from delimited text files. Sadly, even Alventis finds itself in this category. For such applications you can export the entire Memo text as a single long line by stripping line breaks from it. Other apps may have trouble with Tab character embedded in Memo text – Strip Tabs option may help those. Finally, even Microsoft Excel and Microsoft Access do not share the same exact format when it comes to Memos – and that, despite being members of the very same Office suite! This is why you find an option specifically tailored for making the exported file compatible with Excel, while the last option is most generic (and should be compatible with Access).

Data Formats allow you to select how specific types of data will get exported. You can select one of the pre-defined formats or type whatever you want it to be following the common formatting convention.

The Preview displays the first several records exactly as they would get exported using the current settings. It is auto-updated whenever you change any of the above options.

Export to Text Box specifies the filename of the destination file where the data will be saved. You can click the dotted button to browse to an appropriate folder.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

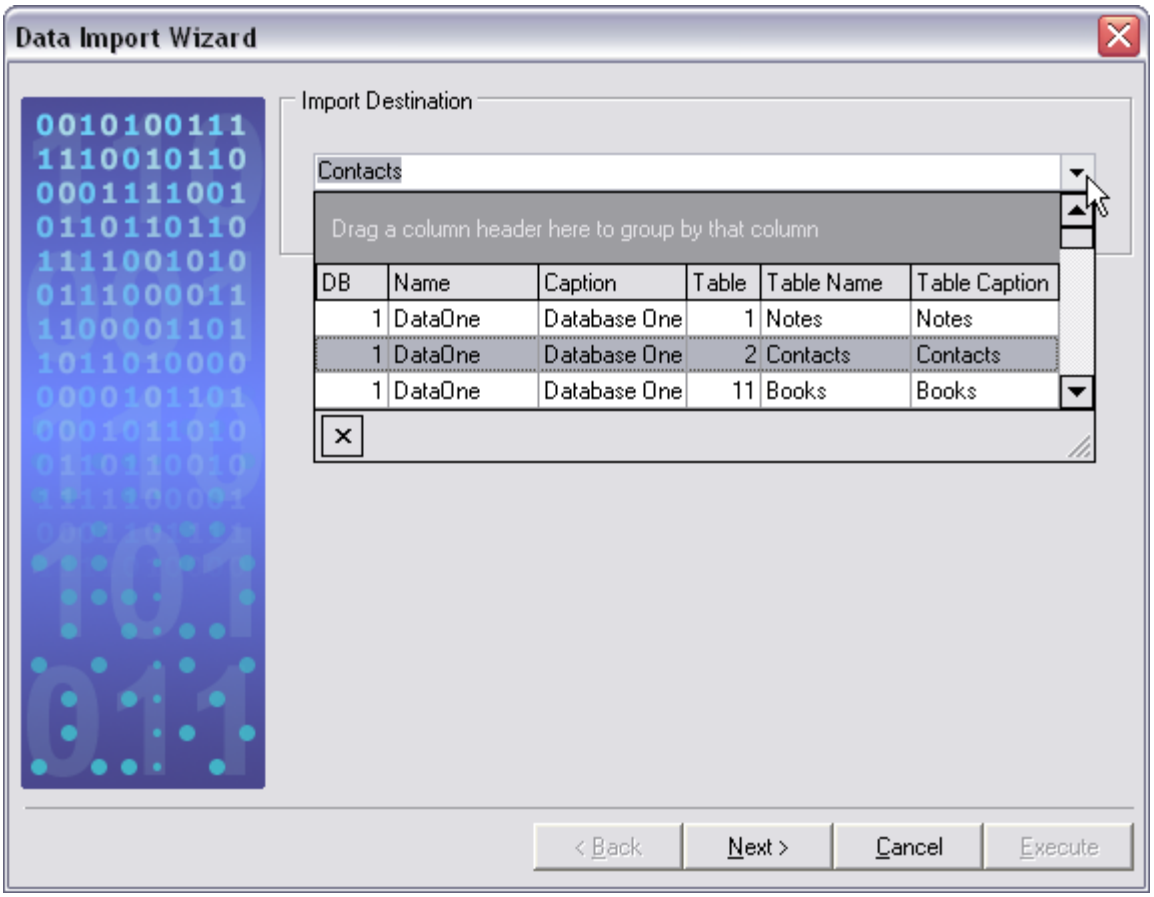
The Export button does the deed.

We have attempted to select the most common defaults for all configuration options in the dialog. Ultimately though, there is no "correct" set of values. Whatever output you end up with, it must be "understood" and correctly imported by whatever application happens to be the "recipient" of the data. If you know that application's import requirements, this should give you a strong indication of what settings you need to use to make it "happy".

Data Import. Clicking the Data Import button launches the Data Import Wizard. It allows you to import data from a variety of formats into an Alventis table of your choice. Unlike other one-step single-screen Wizards used in Alventis, this one is a "real" multi-step beast of a Wizard. The bad news is that there are so many screens and options that explaining them all in detail would double the size of this Guide. The good news is that it is fairly intuitive, so you should be able to find your way through it quite easily. We will therefore mention only some selected facts and issues here and leave the rest to your intuition.

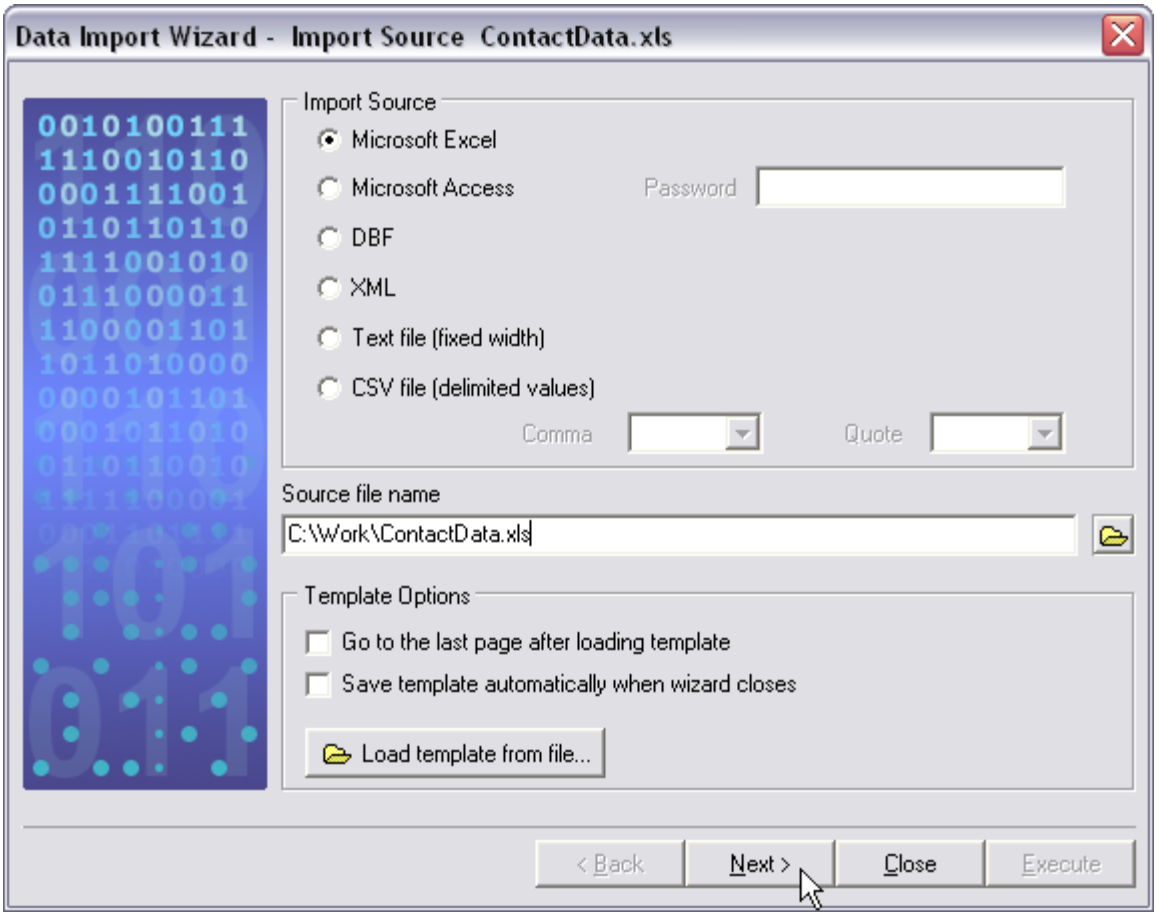
Before we go any further, you should remember that an Import operation will modify your local data. If you ever make a mistake, such modification can lead to drastic results, including data loss. You should therefore have a secure recent backup of your data before you attempt importing something into it.

The Wizard starts by asking you for an Import Destination, which is of course the Database and Table into which you would like to import some data.



Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Once you have selected the desired Destination, clicking Next brings you to the Import Source page of the Wizard depicted below:

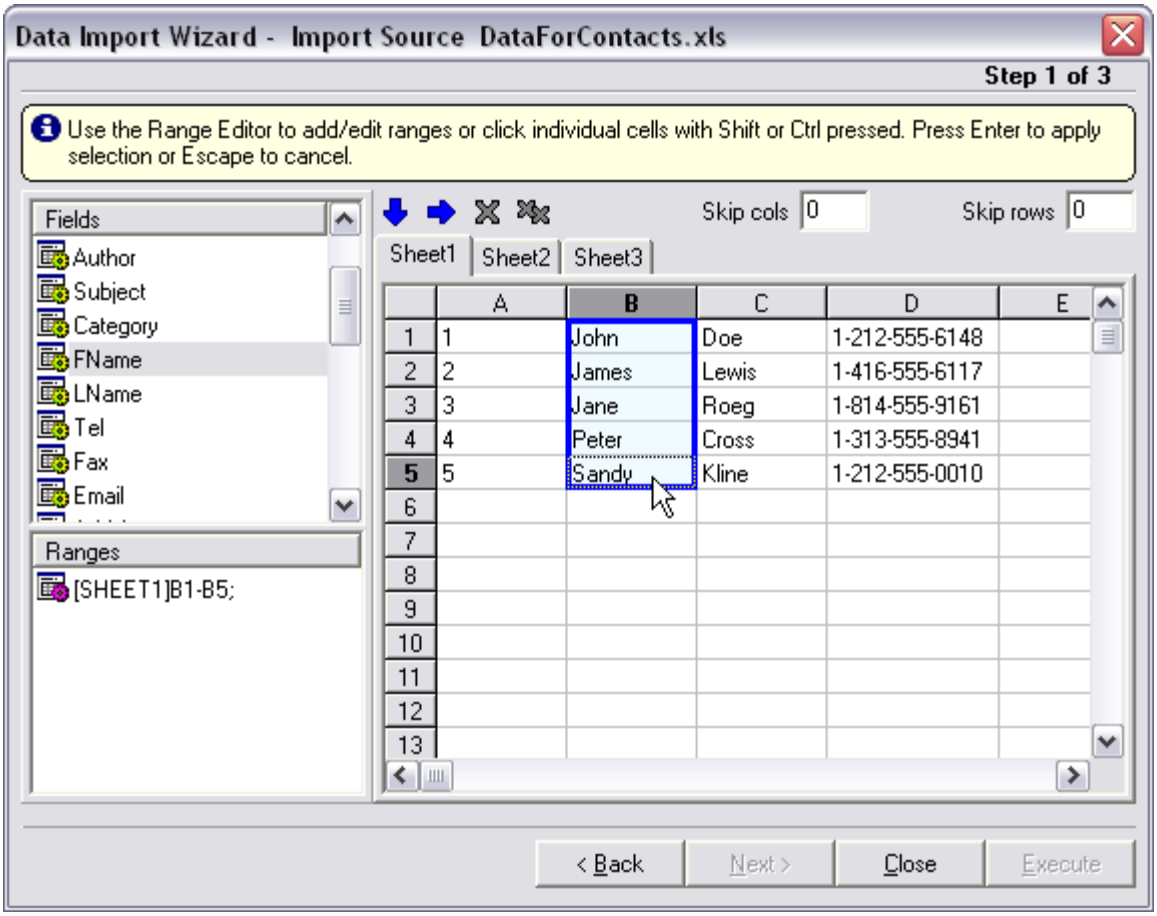


You are presented with a choice of 6 import formats:
- Microsoft Excel (spreadsheet or workbook)
- Microsoft Access database
- DBF files
- XML (Extensible Markup Language) files
- Fixed-width Text files
- CSV files with delimited values

The Template that the lower portion of the dialog refers to is simply a saved import setup. You can save and re-load such setup Templates if you frequently make the same kind of import.

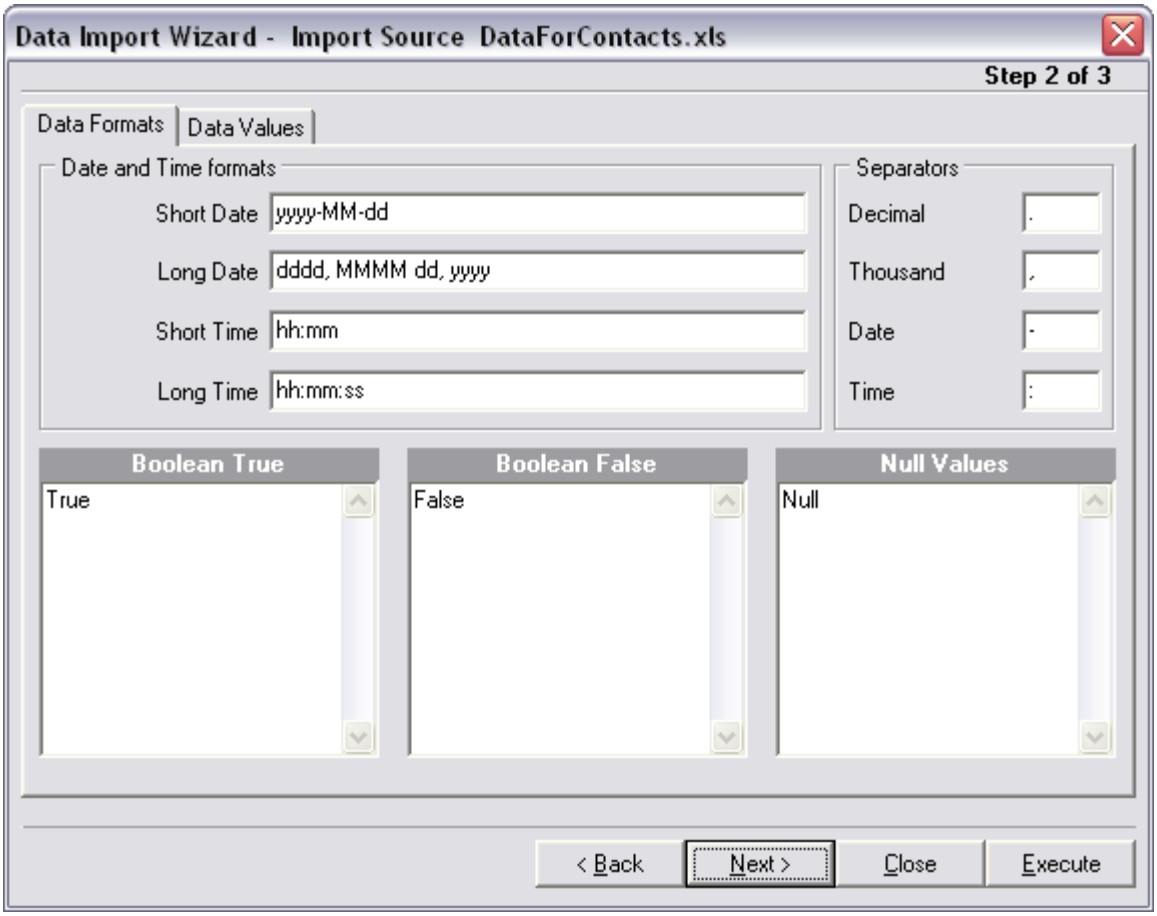
The next step of the Wizard will depend on which input format you have selected. The basic task you will be presented with is explaining to the Wizard what incoming data should go into what fields of the Destination table. The snapshot below shows the interface you would use when importing from an Excel spreadsheet.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Once this is done, you arrive to the next screen where you can specify the Data Formats that will be used to interpret the incoming data, when necessary.

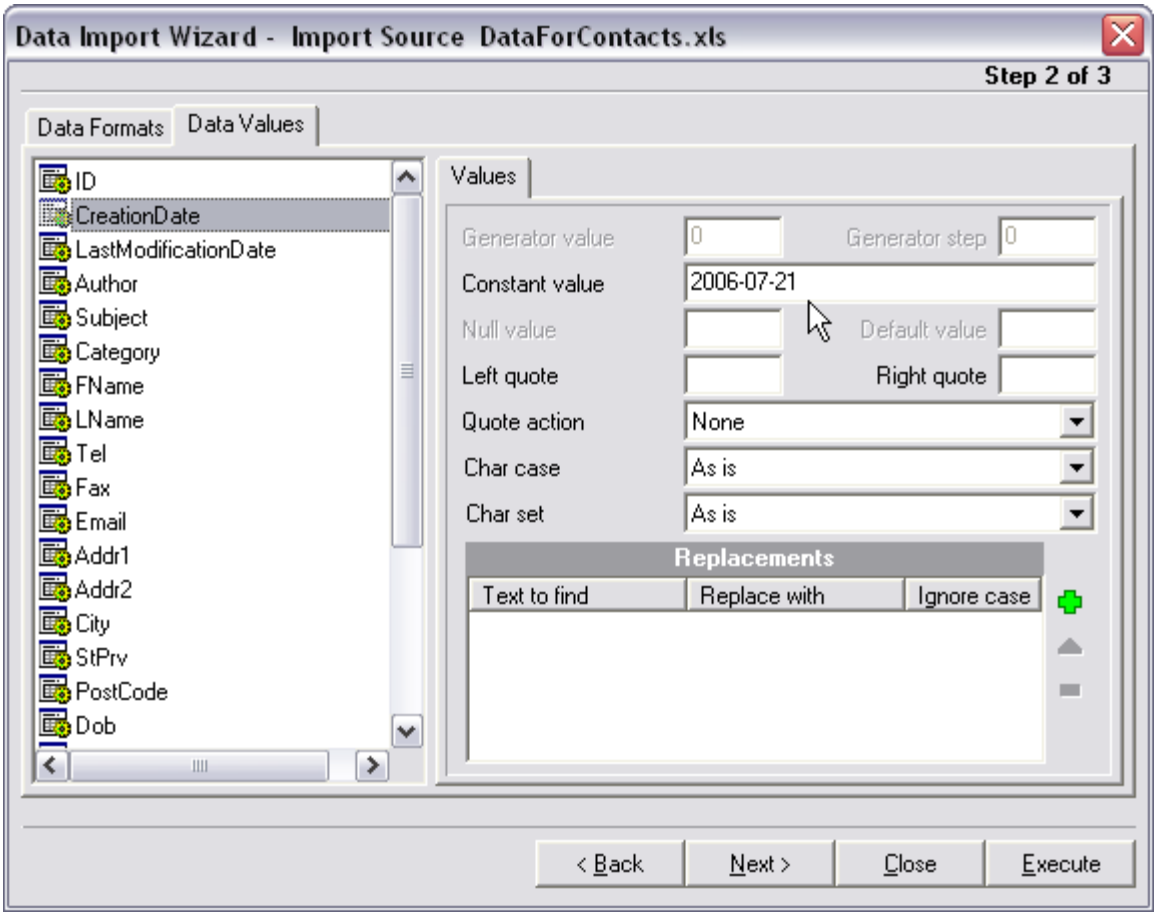
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Data Values section lets you make some adjustments to how the destination fields are filled out. You can, for example, auto-fill some field's values using the Generator value/step controls. Each imported record will have the specified values placed in the field of your choice. For example, if you were to set Generator value/step to 10 and 5 respectively for field SomeNumber of the Destination table, the imported records would have this field's values set to: 15, 20, 25, 30, etc. All of this is specified on a per-field basis: select the field on the left, and set whatever options you want for it, if any, on the right.

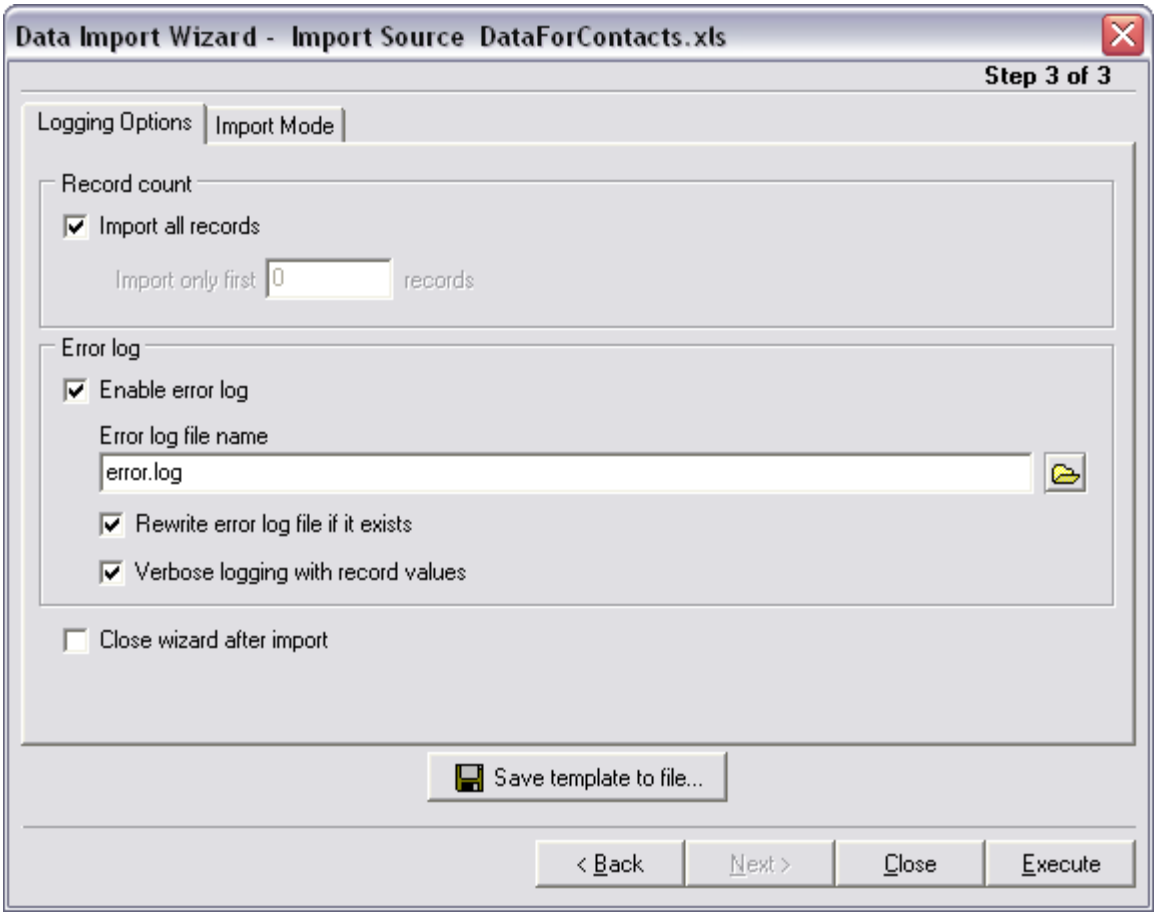
You can specify what value you want in place of Null values or what quote characters you want to insert. You can even perform automatic search-and-replace operations on the incoming data.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The final, 3rd Step of the Wizard lets you specify how many records you want to import, as well as some logging options. You can save the error log (should there be any import errors) to a file of your choice. If you elect to not save it to a file, you will be presented with a list of errors on-screen.

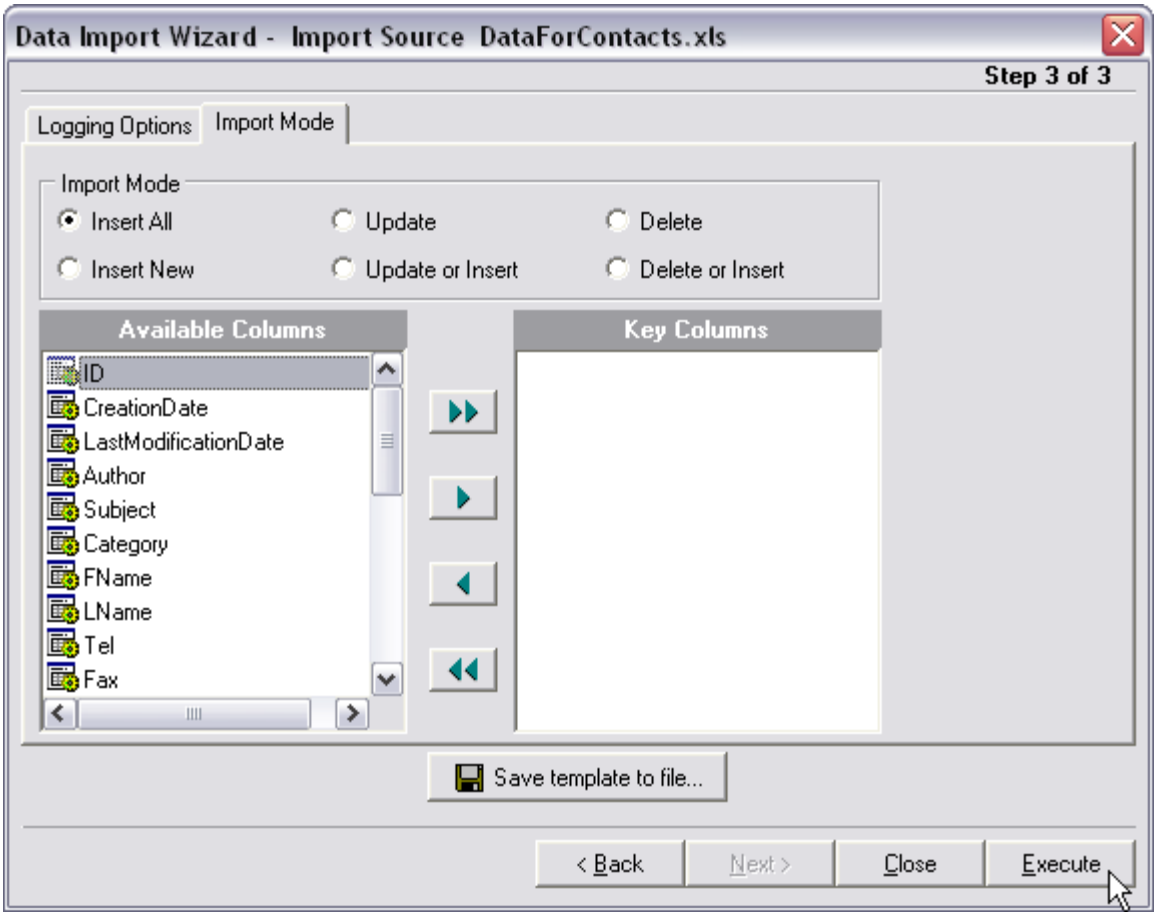
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Import Mode tab sheet may be the most important and complicated step. You should read our general discussion of InfoSet Import Modes for a conceptual understanding of what is going on and what possibilities exist. Unlike the InfoSet Import which offers you a choice of only the Modes that Alventis has determined as valid and reasonable, the Data Import Wizard always offers you the full spectrum of options. Whatever records it does not manage to import (most probably due to Key violations: two records with the same ID value), it will simply list in the error log/list you will be presented with.

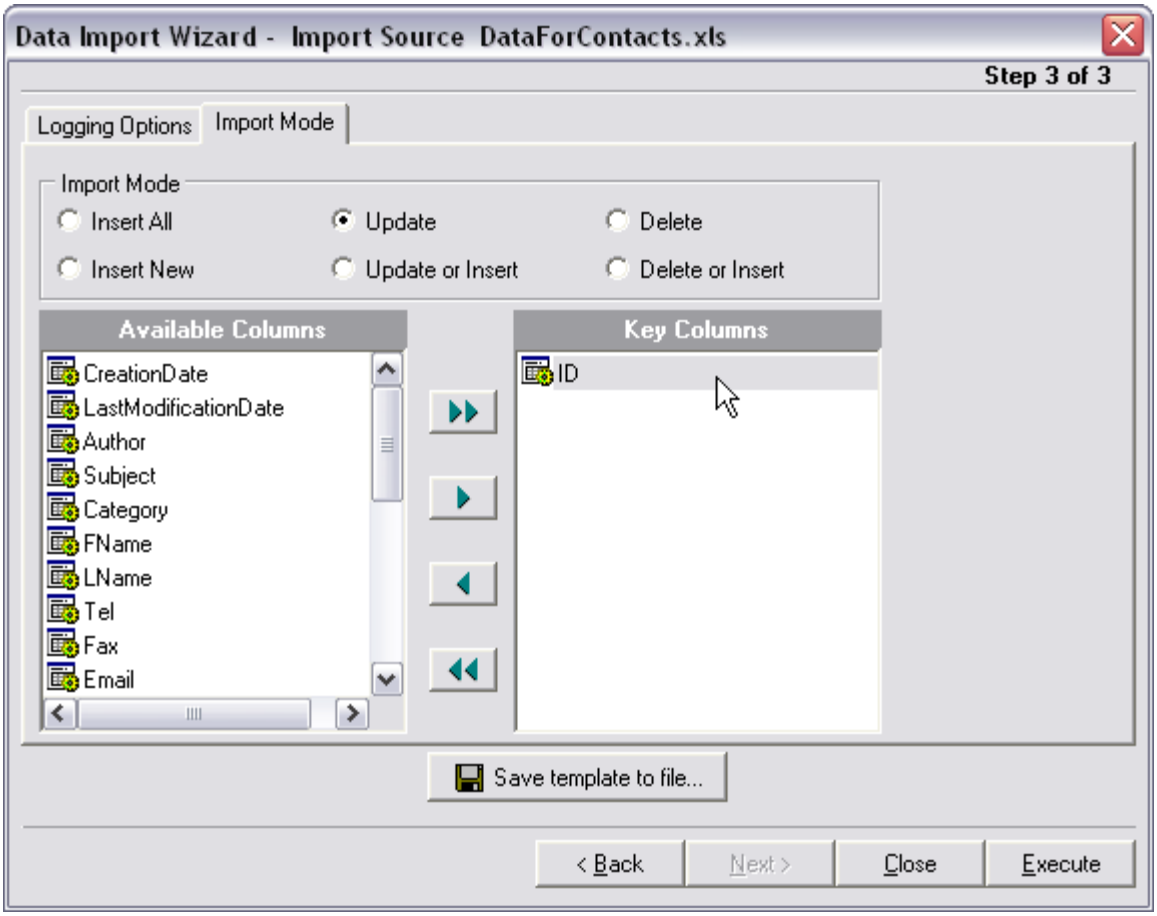
- Insert All will attempt to import all incoming records.
- Insert New will import only unmatching ones.
- Update will only import matching ones.
- Update or Insert will update matching records and import unmatching ones.
- Delete will delete matching local records.
- Delete or Insert will do the same plus import unmatching records.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Key Columns would normally be set to just the ID field since this is the key of all Alventis tables. This is used by some of the above Import Modes to figure out which records are matching and which aren't.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



It should be noted that you can import multi-line plain-text Memo fields – but only from Microsoft Access databases.

2.22.3 Memo Import/Export

Alventis Memos are – like all other table fields – saved in the corresponding data table and loaded from it. You may occasionally find a need to save the contents of a Memo to a disk file or insert the contents from some file into a Memo. These two operations are discussed below.

Memo Export is conceptually similar to a "Save As" command you may be familiar with from other programs. This is exactly how it works in Alventis too. Clicking the Memo Export button opens the familiar "Save As" dialog in which you pick the format of the file in which you want to save the Memo and the destination folder. Alventis supports the following file formats:

- RTF / Rich Text Format. This format is supported directly by Alventis and should preserve most Memo formatting quite well.
- HTML and HTML with CSS are two "flavors" of supported HTML output. They differ in how faithfully they maintain various aspects of document formatting, so you may want to experiment with both to see what works best for you.
- Text Files. The Memo will have its text exported to a plain old text file of your choice. All formatting

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

will be lost of course.
- Unicode Text Files. Same, but the resulting text file will use the Unicode format.
- Microsoft Office Converters. The rest of the export format list will vary depending on your system setup, specifically, on whether you have any Microsoft Office Converters installed on your machine. These are usually installed if you have Microsoft Office. If so, Alventis will attempt to use them to support whatever formats they support. These would typically include various flavors of Microsoft Word and Works, and Corel WordPerfect.

Memo Import is just as simple. Clicking the Memo Import button will open the standard Open dialog from which you must pick the desired file format and file. The imported text will be inserted at the current cursor position in the Memo. The supported file formats are pretty much the same as those supported by Memo Export, but we'll list them again:

- RTF / Rich Text Format. Alventis should be able to import most RTF files with proper formatting. This is definitely the preferred import format.
- HTML. Note that we didn't say "CSS". That's because generally-speaking, HTML with CSS is not supported. Alventis should be able to import HTML with CSS produced by its own Memo Export, but not necessarily any other CSS. You can give it a shot, but don't be upset if the results are not quite right. Import from non-CSS HTML is not stellar either, but if you want to get all the text from a Web page into a Memo, it should work reasonably well. If you save your HTML file in your browser with images (e.g., in Microsoft Internet Explorer: Save As... Web Page, Complete; in Opera: Save As... Web page with images), the HTML file should be loaded with images.
- Text Files and Unicode Text Files. Same as with Export.
- Microsoft Office Converters. The situation here is also similar to Memo Export: if you have them installed on your computer, more import formats will be available. Please note that since Alventis is merely using the available Converters, it has practically no control over their operation, so the success of the conversion may vary greatly, and there's not much we can do about it. You would likely obtain the same result in, say, Microsoft Word if you tried to import the same file into it using the same Converter. Sometimes trying a Converter for a different listed version of the same application may yield better results, so you may need to experiment a bit if the file you are trying to import is not properly decoded right away.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Designer

3 Designer

3.1 Alventis Designer Overview

Alventis Designer is a second application that is part of the Alventis Professional and Enterprise Suites. You may therefore not have Designer. If this is the case, you have two choices:

- Skip all material in this and several subsequent chapters dedicated to Designer, and never find out what you are missing, or
- Read at the very least this introductory chapter to get a better idea of how Alventis works and what "makes it tick".

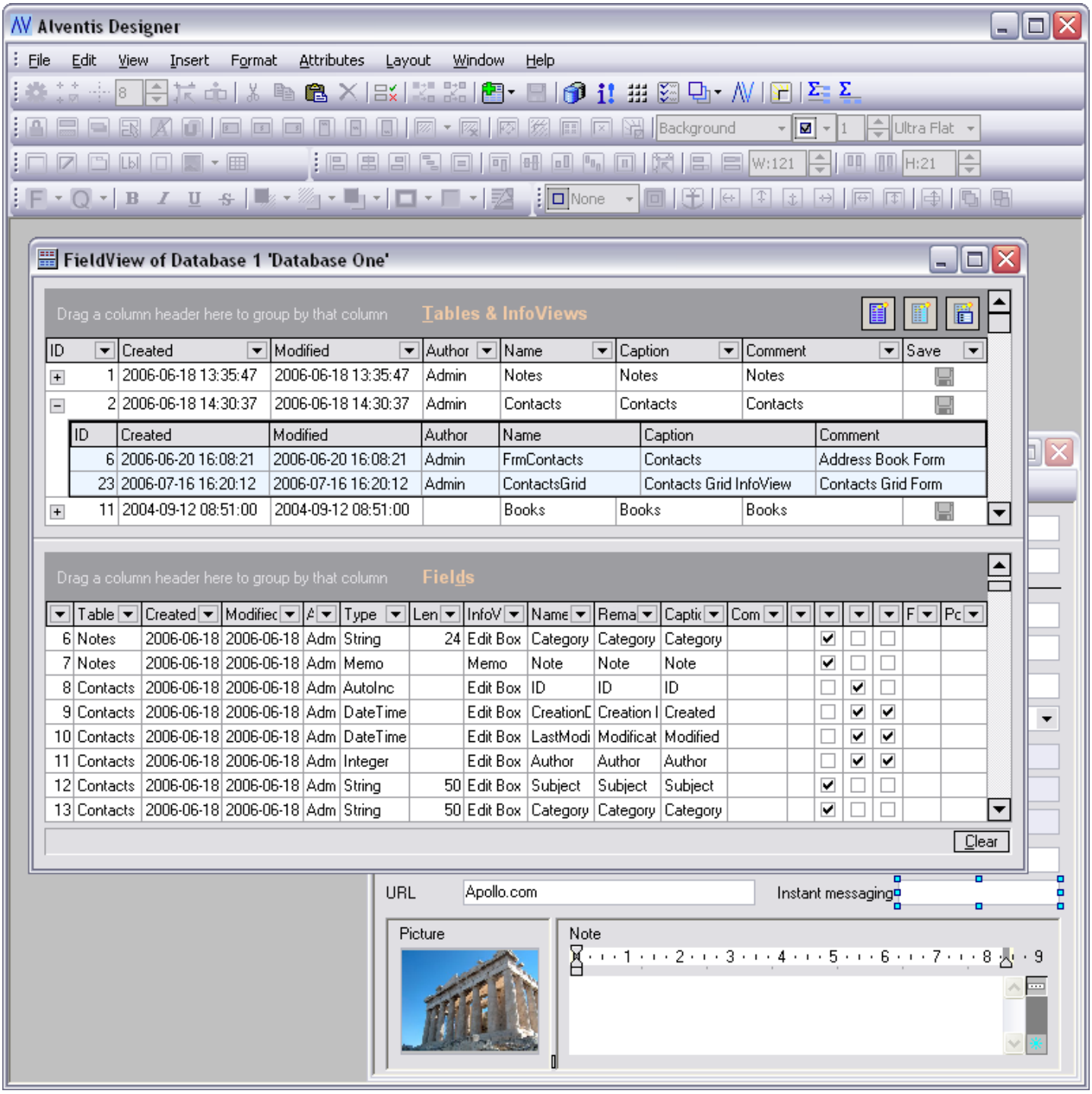
Since this is merely an Overview of what Designer is, and you are undoubtedly eager to "take the plunge" and see what it is really capable of, we'll try to keep it brief and simply outline what Designer is and does.

Designer does basically two "simple" things:

- Create and manipulate data tables
- Create and manipulate associated InfoView forms

That's all there is to it, really. Each of these tasks is, of course, what differentiates Alventis from a myriad of other data-management application.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


Alventis Designer
File Edit View Insert Format Attributes Layout Window Help
FieldView of Database 1 'Database One'
Drag a column header here to group by that column
Tables & InfoViews
ID Created Modified Author Name Caption Comment Save
1 2006-06-18 13:35:47 2006-06-18 13:35:47 Admin Notes Notes Notes
2 2006-06-18 14:30:37 2006-06-18 14:30:37 Admin Contacts Contacts Contacts
ID Created Modified Author Name Caption Comment
6 2006-06-20 16:08:21 2006-06-20 16:08:21 Admin FrmContacts Contacts Address Book Form
23 2006-07-16 16:20:12 2006-07-16 16:20:12 Admin ContactsGrid Contacts Grid InfoView Contacts Grid Form
11 2004-09-12 08:51:00 2004-09-12 08:51:00 Books Books Books
Drag a column header here to group by that column
Fields
Table Created Modifiec A Type Len InfoV Name Rema Capti Com F Pc
6 Notes 2006-06-18 2006-06-18 Adm String 24 Edit Box Category Category Category
7 Notes 2006-06-18 2006-06-18 Adm Memo Memo Note Note Note
8 Contacts 2006-06-18 2006-06-18 Adm AutoInc Edit Box ID ID ID
9 Contacts 2006-06-18 2006-06-18 Adm DateTime Edit Box CreationD Creation I Created
10 Contacts 2006-06-18 2006-06-18 Adm DateTime Edit Box LastModi Modificat Modified
11 Contacts 2006-06-18 2006-06-18 Adm Integer Edit Box Author Author Author
12 Contacts 2006-06-18 2006-06-18 Adm String 50 Edit Box Subject Subject Subject
13 Contacts 2006-06-18 2006-06-18 Adm String 50 Edit Box Category Category Category
Clear
URL
Apollo.com
Instant messaging
Picture
Note

As has been previously mentioned in the Overall Alventis Concept chapter, all Alventis tables have several common System fields, e.g., Creation Date, Last Modification Date, Subject, Category. Everything else is up to you, and Designer allows you to add as many fields as you need to any table. You have full control over the data types (and when appropriate, sizes) of these fields. Things like strings, numbers of all kinds, Memos, and pictures – are all within reach with just a few clicks. We will show you how exactly you create tables and fields in the next chapter.

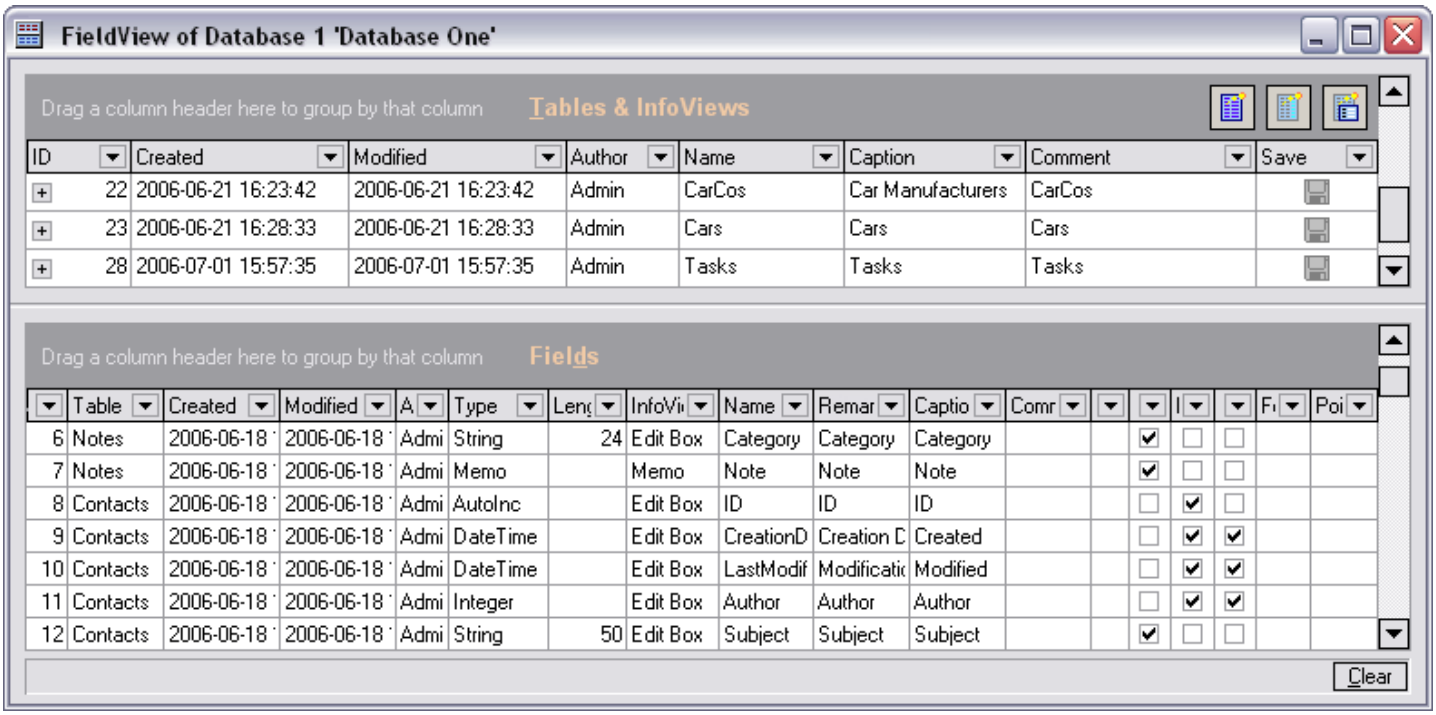
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Once your table structure is ready, creating an InfoView for the table is a snap. This is discussed in the chapter on InfoView Editing, so you'll have to hold your breath for now. Basically, it's mostly drag-and-drop with some formatting that works exactly as if you were formatting text in an Alventis Memo, but that's all we want to say about it for the time being.

Any new InfoSet (a table with its "accompanying" InfoViews, remember?) becomes available to Alventis right away and in exactly the same way as any other InfoSet that may have been shipped with Alventis. There is no difference whatsoever between an InfoSet you create and any other InfoSet in the world – the ones created by us included. You can start by examining the existing InfoSets and what "stuff" they are made of, and soon you'll be on your way to creating your own, hopefully even better and more useful ones. So, without further ado, let's create some tables!

3.2 FieldViews

Designer has just a single main kind of form. This form is called **FieldView** and, as its name suggests, its main task is to manage table fields. There is exactly one FieldView per Database, so everything a FieldView displays automatically belongs to a single Database. Naturally, you can open more than one FieldView concurrently if you want to work on more than one Database at the same time. You can open a FieldView in two ways. Either select the one you want from the dropdown menu opened by the FieldViews button , or double-click the corresponding Database in the Database Explorer form .


FieldView of Database 1 'Database One'
Drag a column header here to group by that column
Tables & InfoViews
ID
Created
Modified
Author
Name
Caption
Comment
Save
22 2006-06-21 16:23:42 2006-06-21 16:23:42 Admin CarCos Car Manufacturers CarCos
23 2006-06-21 16:28:33 2006-06-21 16:28:33 Admin Cars Cars Cars
28 2006-07-01 15:57:35 2006-07-01 15:57:35 Admin Tasks Tasks Tasks
Drag a column header here to group by that column
Fields
Table
Created
Modified
Type
Name
Remar
Captio
Comr
Poi
6 Notes String 24 Edit Box Category Category Category
7 Notes Memo Memo Note Note Note
8 Contacts AutoInc Edit Box ID ID ID
9 Contacts DateTime Edit Box CreationD Creation D Created
10 Contacts DateTime Edit Box LastModif Modificatic Modified
11 Contacts Integer Edit Box Author Author Author
12 Contacts String 50 Edit Box Subject Subject Subject
Clear

The FieldView form is remarkably simple. It is divided in two panes with a Splitter between them. The top pane is occupied by the Tables grid. The lower pane holds the Fields Grid.

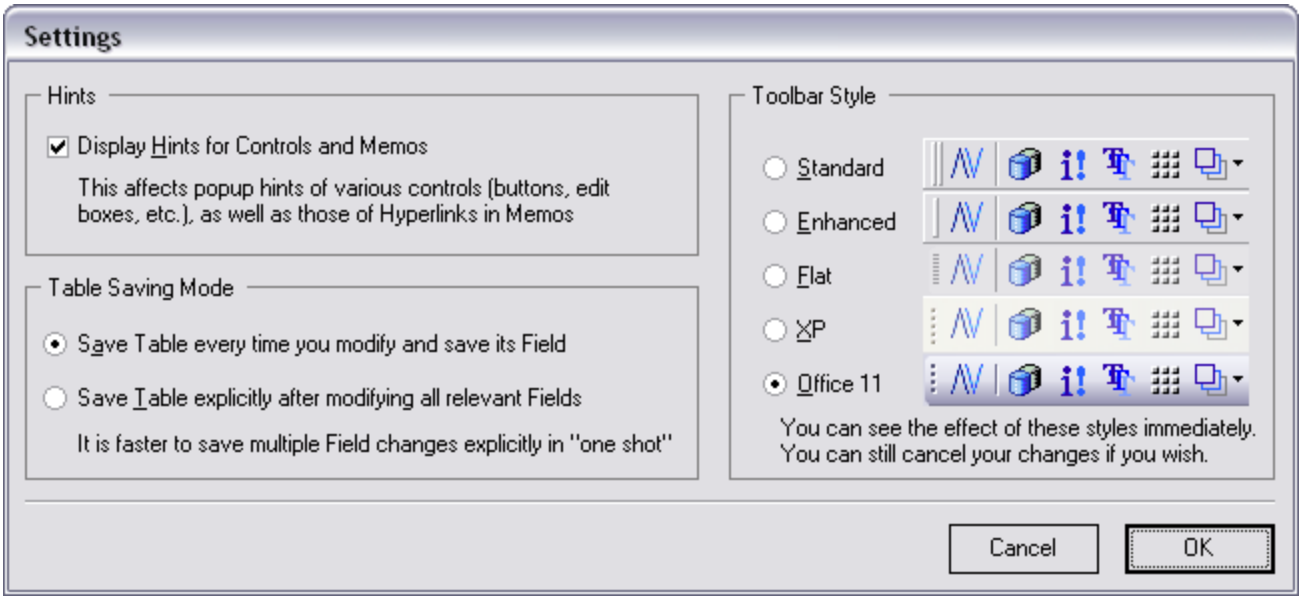
Tables Grid

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

This grid lists all tables of the FieldView's Database. Most of its columns are fairly obvious, so let's quickly go over them:

- ID is the table's ID number, unique index auto-assigned at the time of its creation.
- Created/Modified are the creation and last modification dates.
- Author is the user who created the table.
- Name is the name of the table. This also ends up being used as the physical file name that holds the data table.
- Caption/Comment have no special meaning and you can set them to whatever you find convenient/useful.
- Save is simply a column full of the familiar Save buttons, albeit in a somewhat unusual place. In Designer, you can make modifications to multiple tables but postpone saving them to the actual physical tables until later. This can give you an opportunity to review your changes and perhaps make some corrections. Any tables that have not been saved will have their corresponding Save buttons enabled.

The above describes Designer's one possible behavior. The alternative is: Designer will save each table every time you make a modification to its field. This does save you the effort of clicking the Save button. There's a subtler benefit to this mode of operation though. In this mode, you can rename existing fields that are not already found on any InfoView forms. The downside is that table saving may be lengthy for a table with thousands of records, so you may prefer to "accumulate" all field modifications and save only once rather than patiently waiting after each and every field modification. Note that not all field editing results in changes to the physical table. You can specify which Designer's Table Saving Mode you prefer from the Designer's Settings dialog invoked using the Settings command . The remaining options this dialog offers are so self-explanatory, we won't bother with them.


Settings
Hints
Display Hints for Controls and Memos
This affects popup hints of various controls (buttons, edit boxes, etc.), as well as those of Hyperlinks in Memos
Table Saving Mode
Save Table every time you modify and save its Field
Save Table explicitly after modifying all relevant Fields
It is faster to save multiple Field changes explicitly in "one shot"
Toolbar Style
Standard
Enhanced
Flat
XP
Office 11
You can see the effect of these styles immediately.
You can still cancel your changes if you wish.
Cancel
OK

The Wizards. In the top-right corner of the Tables grid you will find three buttons. They invoke Wizards that should assist you in creating the three types of objects the FieldView is "responsible" for:

New Table

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

New Field

New InfoView

The Wizards are quite intuitive and self-explanatory, so we shall not explain them in detail. You can also invoke these Wizards from the File menu. It should be noted that the Wizards perform the exact same tasks that you can accomplish by simply inserting a new record into the corresponding grid – directly in the FieldView form, as explained below.

- You can create a new table by simply creating a new record in the Tables grid: with this grid focused, hit Insert on the keyboard, specify the table's Name (the only important value!) and you're done.
- You can modify an existing table's Caption and Comment by editing the respective values.
- You can also delete a table by deleting its record. As a safety measure, you cannot delete a table that has InfoView forms based on it (which would usually be the case). If you absolutely insist on deleting it, you have to delete the InfoViews first, and only then go after the table. You will obviously lose the table's structure and all its data, so be careful what you wish for.

This would be the end of our discussion of the Tables grid – if it weren't for the little Plus buttons in the ID column. Yes, this is a two-level grid. The top level holds the list of tables that we have just examined, while the second level holds the InfoView forms of each table. We'll get to them in the next chapter, so let's pretend for now that they are not there.

Fields Grid

This grid lists all fields of all tables from the Database.

Drag a column header here to group by that column Fields

ID	Table	Created	Modified	Author	Type	Length	InfoView Item	Name	Remark	Caption	Comment	Default Value	Text-Indexed	Read-Only	Required	Formula	Pointer Target
177	Budget	2006-07-02	2006-07-02	Admin	AutoInc		Edit Box	ID	ID	ID			☐	☑	☐		
178	Budget	2006-07-02	2006-07-02	Admin	DateTime		Edit Box	CreationDate	Creation Date	Created			☐	☑	☑		
179	Budget	2006-07-02	2006-07-02	Admin	DateTime		Edit Box	LastModificationDate	Modification Date	Modified			☐	☑	☑		
180	Budget	2006-07-02	2006-07-02	Admin	Integer		Edit Box	Author	Author	Author			☐	☑	☑		
181	Budget	2006-07-02	2006-07-02	Admin	String	50	Edit Box	Subject	Subject	Subject			☑	☐	☐		
182	Budget	2006-07-02	2006-07-02	Admin	String	50	Edit Box	Category	Category	Category			☑	☐	☐		
183	Budget	2006-07-02	2006-07-02	Admin	Currency		Calculator Edit Bc	Amount	Amount	Amount			☐	☐	☐		
184	Budget	2006-07-02	2006-07-02	Admin	Date		Date Edit Box	Date	Date	Date			☐	☐	☐		
185	Budget	2006-07-02	2006-07-02	Admin	String	4	Edit Box	DateYear	DateYear	Year			☐	☐	☐	EXTRACT(YEAR FROM Date)	
186	Budget	2006-07-02	2006-07-02	Admin	String	2	Edit Box	DateMonth	DateMonth	Month			☐	☐	☐	EXTRACT(MONTH FROM Dat	
187	Budget	2006-07-02	2006-07-02	Admin	String	6	Edit Box	Credit	Credit	Credit			☐	☐	☐	IF(Amount < 0, '-', '+')	
188	Budget	2006-07-02	2006-07-02	Admin	Boolean		Check Box	TaxDed	TaxDed	Tax-deductible			☐	☐	☐		

All fields are listed together so that you could easily find all fields by their data type (e.g., all Picture fields) or all fields that have some other feature in common. Whenever you find yourself working with a single table, you may want to reduce the clutter by simply either grouping this grid by Table or filtering on the Table column.

Whenever the Fields Grid is sorted primarily by the table, you can use drag-ordering to place a table's fields in the order you find convenient. This display order of fields is purely visual and has no effect on anything other than the Fields Grid. When the Fields Grid displays fields from multiple tables, you have to be careful to only drag-order fields of a table within that table: dragging a field outside of the range of fields belonging to the same table (or even to its top or bottom) will result in the field ending up in unexpected places. Drag-ordering therefore works best when the Fields Grid is filtered to display fields from only a single table.

Following is a list of the available columns and their descriptions.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

- ID is the usual auto-incremented field ID.
- Table is the table the field belongs to.
- Created/Modified are the usual dates.
- Author is obviously the creator of the field.
- Type is the data type of the field, e.g., Integer, String, etc. We'll be discussing these shortly.
- Length is the length of the field. This is only meaningful for String fields. More on this later.
- InfoView Item offers you a choice of what Control to represent the field with on InfoView forms. We'll talk about it in a bit.
- Name is the name of the field. This is what goes into the actual physical table structure. You may not be able to change it for any fields referenced from any InfoView forms, so choose field names carefully. They must start with a letter and must only contain alphanumeric characters or the underscore.
- Remark is similar to Name, except that the Remark is for your information only: it is not used by Alventis, so you can set it to whatever you find meaningful. You are free to use whatever characters you like here.
- Caption will be used, where appropriate, as the default Caption for the Control representing the field on the InfoView form. You can edit all Control's Captions later, but you have an opportunity here to specify a sensible default value in case you end up "implementing" the same field more than once, perhaps on multiple InfoViews.
- Comment is just a comment you want to associate with the field. Except that this Comment is saved in the physical table structure and it is used under some circumstances as a replacement for the field's Name in Captions (for instance, it may get used by grids displaying records from the table). You should therefore leave it empty unless you know what the consequences are and when/where they will manifest themselves.
- Default Value is the default value you'd like this field to get assigned when creating new records. You must specify values that make sense for the data type of the field, e.g., an Integer field may have a Default value of *25* but not *Xyz*.
- Text-Indexed, when checked, specifies that the field's contents should be indexed using Alventis' full-text indexing, so that searches performed in UniGrids would take its contents into consideration. By default, all String and Memo fields are indexed in this fashion, so Alventis searches through all your textual data. You can disable full-text indexing of some fields if you do not want them to be searched for some reason.
- Read-Only, when checked, specifies that the field is read-only and should not be editable by the user. Some standard System fields are read-only due to their nature, for example. Setting this flag to true may only make sense for your own field if it has already received all the values you want, so no further editing should be possible. This sounds like a far-fetched scenarios though, so you'd typically want to leave this checkbox in its default, unchecked state. You can always make read-only the Control that implements this field on a specific InfoView, which would probably make much more sense.
- Required, when checked, specifies that this field must be assigned a value for a record to be posted to the table. An attempt to post a record with this field left empty will generate an error that the user will need to correct. You can use this to ensure that the most crucial information is never left blank.
- Formula is used for Calculated Fields. This is a fairly advanced topic, and we'll discuss it in the Calculated Fields chapter.
- Pointer Target column is only applicable to Pointer type fields that are used to link Relational tables. We'll be looking at them in more detail at a later time.

Let's go back and examine some columns we have mentioned in more detail.

Data Types. Designer supports the following field data types:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

String	Text of the specified Size (1 to 512 characters)
Small Integer	16-bit integer (-32,768 – 32,767)
Word	16-bit unsigned integer (0 – 65,535)
Integer	32-bit integer (-2,147,483,648 – 2,147,483,647)
Large Integer	64-bit integer (up to +/-9,223,372,036,854,775,807)
Float	Floating-point number: positive or negative, 5.0 * 10^-324 to 1.7 * 10^308 with an accuracy of 15 digits
Currency	Floating-point monetary amount number
Auto-Increment	Auto-incrementing 32-bit integer counter
Boolean	Boolean (True/False)
Date	Date
Time	Time
Date/Time	Timestamp, i.e., both Date and Time
Memo	Alventis Memo
Graphic	Picture/image
Pointer	32-bit integer used as foreign key

We'll be talking about some of the above Types in the following sections. Changing a field's Type may result in data loss, so you must be extremely careful (and have a recent backup – which is something you should have in any case). Some field conversions work with no data loss. For example, converting a String field that holds properly-formatted integers to an Integer field should preserve the existing values. Most field types should convert to String type with few problems. Beyond that, you should exercise caution. Same goes for String field's Length. Extending the Length of an existing String field will always preserve existing data. Conversely, reducing its Length will preserve whatever fits, but may obviously "chop-off" long text values.

InfoView Item. Designer uses the following principle to simplify your task of creating InfoView forms. Each field of a table is assigned a particular InfoView Item, also referred to as Control. This is the widget or thingy that will appear in the InfoView and will be used to display and edit the field's value. We could say that a Control "implements" a field, is attached to it, and so on. Designer therefore establishes an essentially one-to-one relationship between fields and Controls. This way, when you want to "place a field on the InfoView", Designer knows what actual Control attached to that field it needs to place on the form. This is why every field must have a Control associated with it. For some field Types you have a choice among several suitable Controls. For others, e.g., Graphic or Memo, there's but a single one.

Following is a list of field Types together with what Controls may be used to represent each of them.

String	Edit Box Radio Button Group Immediate Lookup Combo Multi-Line Edit Box

Small Integer Word Integer	Edit Box Radio Button Group Immediate Lookup Combo SpinEdit Calculator Edit Box

Float	Edit Box SpinEdit

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

	Calculator Edit Box Currency Edit Box
Currency	Edit Box Calculator Edit Box Currency Edit Box
Auto-Increment Large Integer	Edit Box
Boolean	Checkbox Radio Button Group Immediate Lookup Combo
Date Date/Time	Date Edit Box
Time	Time Edit Box
Memo	Memo
Graphic	Image
Pointer	Lookup Combo Box

We will now take a look at each Control individually.

- Edit Box. This is the most basic of them all: the common Text Box for general-purpose text entry. A natural choice for String fields.
- Multi-Line Edit Box. A plain-text memo which may be suitable for some String fields, especially Calculated or other read-only ones.
- SpinEdit. An editor suitable for numeric fields. You can change its value by "scrolling" it with the mouse.
- Currency Edit Box. Essentially an Edit Box that always displays its contents with your locale's Currency format, e.g., $1,200.00.
- Calculator Edit Box. An Edit Box with a little calculator attached to it. Allows you to enter the value you want either directly or by using the calculator.
- Checkbox. We're sure you know these "squares".
- Date Edit Box. An editor with a dropdown calendar to help you enter a date (which you can key-in directly if you prefer).
- Time Edit Box. Same idea, but you enter the time using a dropdown clock.
- Image. A Picture Box that allows you to display, load, and save images stored in the corresponding field.
- Memo. The opulent Alventis Memo editor with all its richness of formatting.
- Radio Button Group. A Radio Group Control allows you to represent a small set of possible field values as a group of options (each – a Radio Button). The underlying field values may be: any integral numeric type (1/2/3/4), a String ("NY"/"CA"/"TX" field values may be represented by Radio Buttons "New York"/"California"/"Texas", for example), a Boolean (True/False can be expressed as "Yes"/"No", "To be"/"Not to be" or whatever else suits you).
- Immediate Lookup Combo. Same exact idea as Radio Group but implemented as a combo box. The functionality and applicability is the same, only the actual interface of the Control is different.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

- Lookup Combo Box. A combo box that displays a grid of records from another table, one referenced by (or pointed to) by the field this Lookup beast of a combo box represents. We will get to know it better when we talk about Relationality.

There may be times when you would want to represent a field using different Controls. For example, you may be creating two InfoView forms for the same table. One could be representing an integer field as a Radio Button Group, while the other could benefit from using an Immediate Lookup Combo in its place. Whatever the case may be, it's quite easy to implement the same field using more than one Control that it supports. Just edit the field changing its Control value to whatever you wish to create at that particular moment in time – and off you go. We'll talk much more about actually placing Controls on InfoViews later though.

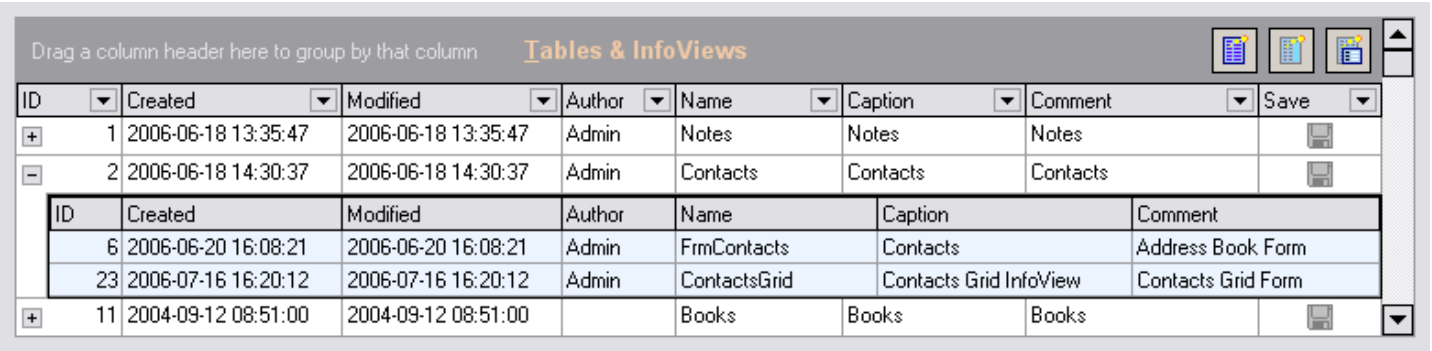
As soon as you create a new table (in the Tables grid), its standard System fields are automatically created. There are very few things you can modify about those, and Designer won't let you alter things you shouldn't meddle with. You may change the length of the system String fields and disable full-text indexing for them. You can also change the Control, Title, Caption, and Comment. With your own fields, you have no limitations.

You can create new fields and delete existing (non-System) ones directly in the Fields grid – as you'd do with any grid record. The Fields grid tries to help you as much as it can. For example, the Type you select for your field influences the behavior of many of the remaining columns: some will offer suitable defaults, others won't let you edit them if this is not appropriate for the selected Type, etc. In all cases, if you can't seem to be able to edit a certain cell's value, this is because you shouldn't. In most cases, the grid will tell you why in the Status bar at the very bottom of the FieldView form. You can clear the message at any time once you no longer need it by clicking the Clear button on the right.

3.3 InfoView Manipulation Overview

We have seen how easy it is to create tables and their fields using the two respective grids of the FieldView form. There is one more thing this form lets us create though.

As we have mentioned earlier, the Tables grid is a two-level grid. The top level lists tables, while the second level lists **InfoView** forms belonging to each table.



ID	Created	Modified	Author	Name	Caption	Comment	Save
1	2006-06-18 13:35:47	2006-06-18 13:35:47	Admin	Notes	Notes	Notes	
2	2006-06-18 14:30:37	2006-06-18 14:30:37	Admin	Contacts	Contacts	Contacts	

ID	Created	Modified	Author	Name	Caption	Comment
6	2006-06-20 16:08:21	2006-06-20 16:08:21	Admin	FrmContacts	Contacts	Address Book Form
23	2006-07-16 16:20:12	2006-07-16 16:20:12	Admin	ContactsGrid	Contacts Grid InfoView	Contacts Grid Form

11	2004-09-12 08:51:00	2004-09-12 08:51:00		Books	Books	Books	

Each InfoView in Alventis is considered to "belong" to its respective main table. Relational InfoViews may "bring together" data from more than a single table, but one of these tables will be considered "main", so even these complex InfoViews can still be deemed attached to a single table. We will be talking about Relational tables and InfoViews in another chapter, but we just wanted to make sure you

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

understand the overall relationship between tables and InfoViews.

Any table may have more than one InfoViews (or none at all, of course, but that would be a little sad). All InfoViews of a table are listed in the second-level sub-grids underneath the table. The columns of these InfoView grids are almost entirely self-evident, so we'll rush through them quickly:

- ID is the auto-incrementing unique index, sometimes also referred to as SpecID.
- Created/Modified/Author are the usual elements.
- Name is the unique name of the InfoView used mostly for your reference.
- Caption is more "visible" since it determines the Title of the InfoView form. You can edit it even for an existing InfoView, whether open or closed.
- Comment can hold whatever text/note you may find useful.

You create a new InfoView – you probably guessed by now – by creating a new record in the grid. Delete the InfoView by deleting the record. Nothing to it.

Once you have created a new InfoView record, you may open the corresponding InfoView form by double-clicking the record or hitting Ctrl-Enter on the keyboard. The new blank form with the Caption you have specified is created for you. You can now put some fields and other interesting stuff onto it, but this is the subject of the next chapter.

It may be a bit premature to tell you this, but it seems most appropriate at this point, so we'll tell you now. An empty InfoView you just created is truly "very empty". You may have created it under the Contacts table, for example, but in its empty state, it has nothing that "bonds" it to this particular table. It is only when you start placing fields onto the InfoView (well, creating Controls corresponding to some fields, really, but we'll allow ourselves to use such imprecise language), so, it is only at that point, that there's something in the InfoView that really establishes a strong link between it and some table. Therefore, whichever field you place first on an InfoView determines what table this InfoView will be "representing" – that of the field you just placed. Usually, you'd obviously place the fields of the table for which you created the new InfoView onto it, so all of the above won't be an issue. But if you make a mistake or just change your mind, and start with a field of some other table, the InfoView will get re-assigned to that other table. This may only happen to an empty InfoView with no existing fields, so InfoViews won't be "jumping around" too much.

3.4 InfoView Editing

In the preceding chapter we got a rough idea of how you create an **InfoView**. Here, we'll see what you do with it once it has been created.

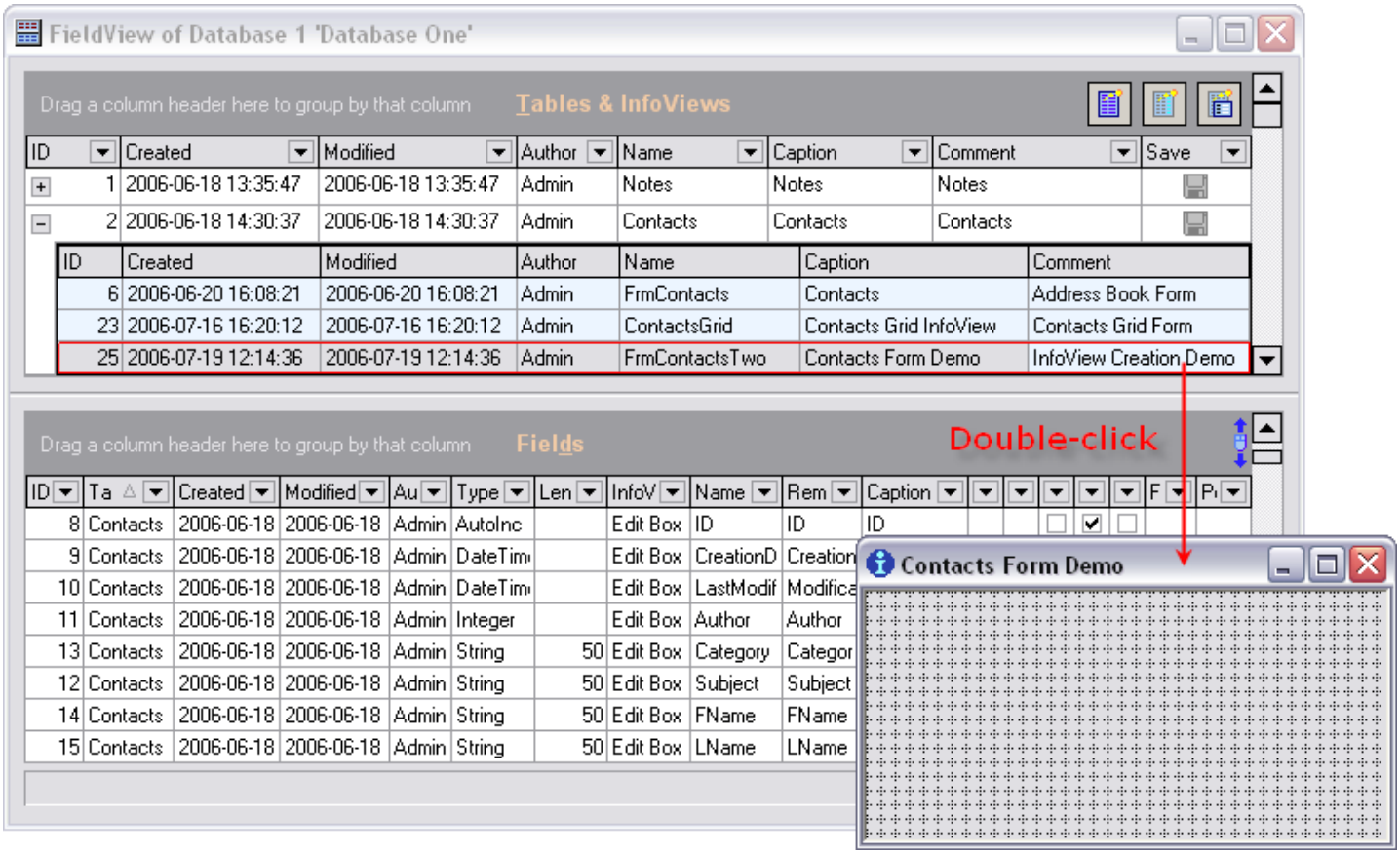
The basic operations are actually quite simple:

- Create new items of various kinds
- Adjust the items on the InfoView form

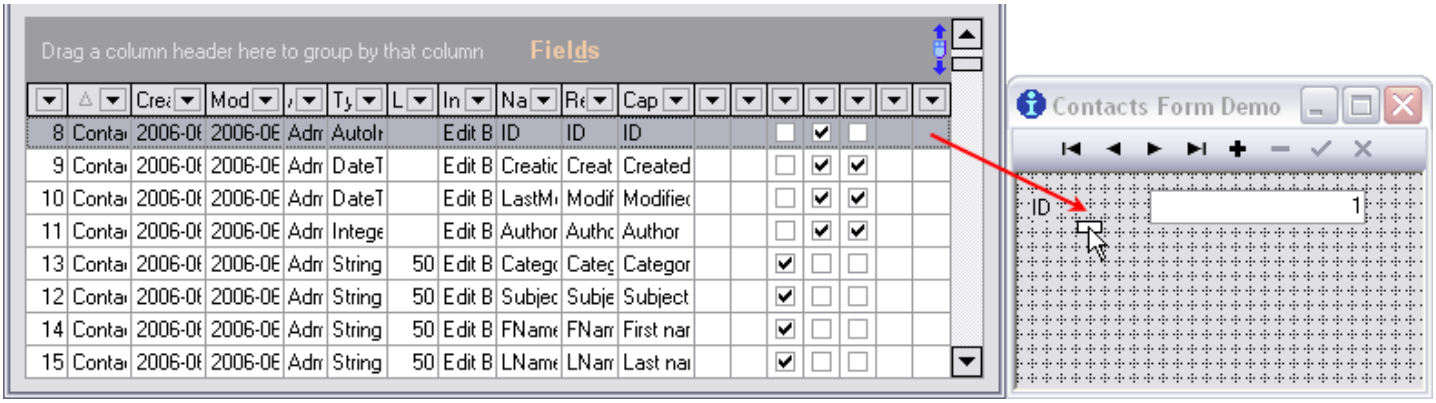
Let's start by creating the most basic but nonetheless functional InfoView to demonstrate at least some of the above. For the following example, we'll be basing our InfoView on the Contacts table, but any table would do just fine.

First, we are assuming that a new InfoView record has been created in the InfoView sub-grid under the Contacts table (as described in the previous chapter). Let's open the new blank InfoView form by double-clicking its record, and let's place it wherever it won't be obstructed by the FieldView form.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


FieldView of Database 1 'Database One'
Drag a column header here to group by that column
Tables & InfoViews
Fields
Double-click
Contacts Form Demo

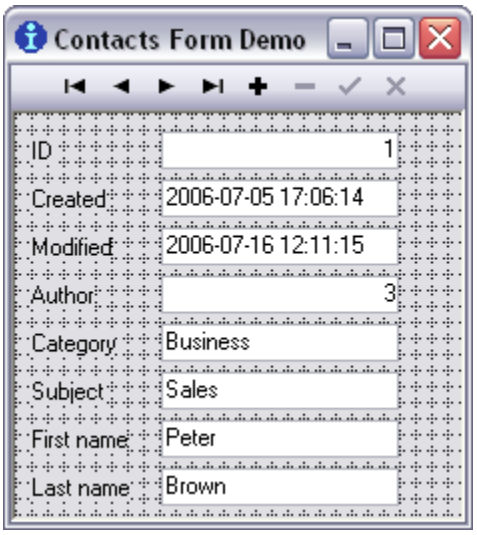
To create a minimally useful InfoView, we must place some fields on it. Placing a field on an InfoView form is an easy-to-follow 14-step process. Just kidding. It's a one-step process, of course. Drag the field you want to place onto the InfoView and drop it roughly where you want it. That's all! You can also double-click the field in the Fields grid – and it will be added to the bottom of whatever may already be in the InfoView. Whichever way you prefer, it's a snap. As soon as you place the first field on the InfoView (let's start with the Contacts table's ID field just to follow some sequence), the InfoView starts to look like... well, like a real InfoView:



You got the DataNav bar at the top plus whatever Control happened to correspond to the field you just added to the form, possibly accompanied by a Caption label.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Let's add the rest of the fields, or at least some of them to keep the ID field company. Drag-and-drop or double-click CreationDate, LastModificationDate, Author, Subject, Category, and whatever else you want: the Controls get added to the form, so it starts to look even more like "the real thing":



At its very simplest, that's it! We have just created a perfectly usable, albeit extremely primitive InfoView. We can save it at any time by clicking the Save InfoView Form button – it will be enabled for InfoViews that have been modified and thus need saving.

Note that the Author edit box doesn't look the same in Designer and Alventis: in the former, it simply displays the User ID, while in the latter, it displays the corresponding User Name. All other items look and work the same in both applications.

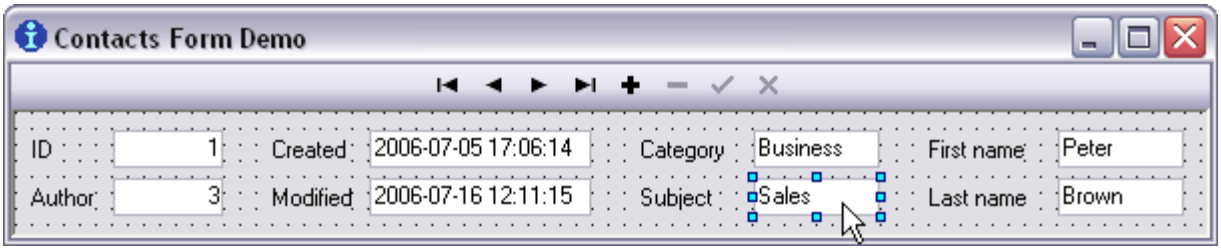
The above procedure for "implementing" fields on an InfoView form may be simple and effective, but there's one other way we'd like to mention before we move on to the fun stuff. You can place all fields of a table onto an InfoView in one shot by dragging the table itself onto the InfoView. Instead of dragging a field from the Fields grid, you'd simply drag the table from the Tables grid – and all fields get created for you one underneath the other. You can then re-arrange them whichever way you like, but this is what the next chapter is about.

An InfoView may be based on a single "main" table. This means, in particular, that you cannot freely mix tables within a single InfoView form: you can only add fields of the table the InfoView is already based on. A field from some other table will not be accepted. Relational InfoViews present all sorts of exceptions to the above rule, but we'll deal with Relationality later.

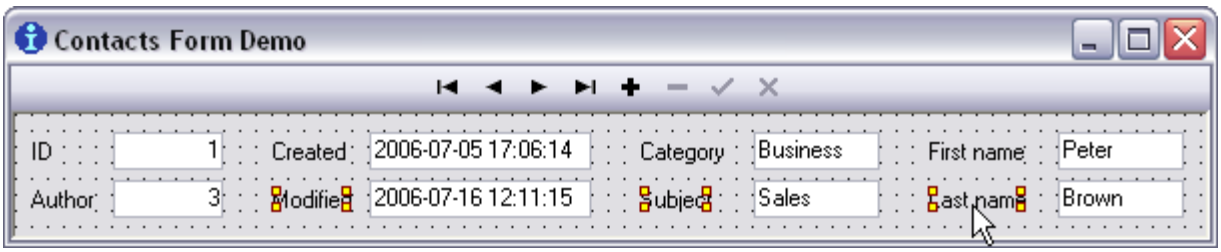
3.5 InfoView Item Selection and Editing

You may be familiar with the concept from some other application. The idea is quite simple. To manipulate any existing items on an InfoView form, you need to select them first, and then do something with them. You select an item by clicking it with the mouse.

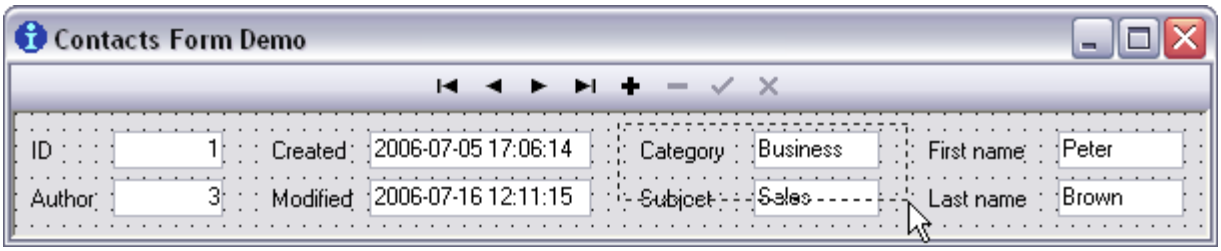
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



You can select more than one item at a time. You add (or subtract) items from an existing selection by Shift-clicking them. Or to put it simply, if you have 4 items A, B, C, D, and you want to select A, B, and D, you'd click A, Shift-click B, and Shift-click D. If you change your mind, and no longer want B to be selected, Shift-click B again.



You can also drag-select items. To do so you must hold down the Ctrl key on the keyboard, and drag across any controls you'd like to select. Doing so draws a dashed selection rectangle as you drag the mouse across the form. Whatever this rectangle touches, becomes selected.



You can combine the two selection methods by drag-selecting some items, and then adjusting the resulting selection if necessary by Shift-clicking some items to either select or deselect them. Just don't forget to hold down Ctrl when drag-selecting.

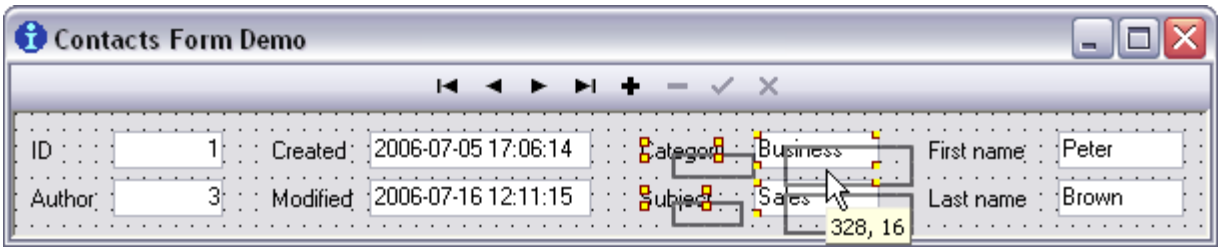
You can also select one item at a time using the keyboard, but this is seldom practical. In case you want to try it, you can press the arrow keys or Tab/Shift-Tab to select next/previous items. The item order corresponds to the order in which they were created, so it may not match their visual arrangement on the form.

One special type of item selection with the Keyboard is quite useful though. Whatever items you may have selected, hitting Escape will select their parent Container. We haven't discussed what they are yet, but remember this fact when we get to them.

To deselect everything, just click on the background of the InfoView.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Moving. One or more selected items can be moved within the boundaries of the InfoView (or whatever other stuff they may be "sitting on" – we'll see what that is later). Just drag the selected controls around with the mouse.



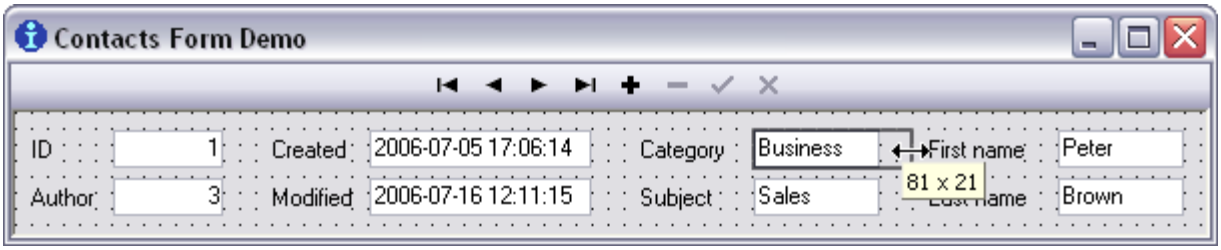
As we mentioned above, you can only move the items within their enclosing "Container". E.g., if you have some items on a Panel, you can move them around within that Panel, but you cannot move them to another Panel. To achieve such a "jump", you would use a Cut-and-Paste operation.

Moving the controls using the keyboard actually works quite well. You use the cursor arrow keys Left/Right/Up/Down together with Ctrl and optionally Shift like so:

- Ctrl-Left will move the selected items 1 pixel to the left (Ctrl-Right – to the right, and so on)
- Ctrl-Shift-Left moves the items one Gridline Step at a time

We'll deal with Gridlines later, but for now, it's just a convenient number of pixels, perhaps 8 or something (and there's a SpinEdit box on the toolbar to control this).

Sizing. You can resize a selected item with the mouse by dragging its sizing handles in the desired direction. The sizing handles are the little squares drawn along the perimeter of the selected item.



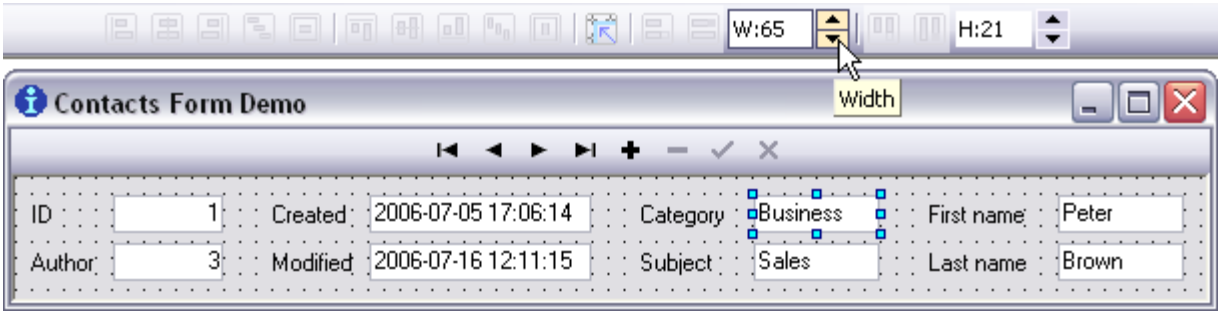
You can also use the keyboard, whose operation is very similar to how you move items with it, except that to resize an item, you use the Shift key (with the same Arrow keys): Shift-Left will shrink the item width 1 pixel at a time, Shift-Right will expand it, etc. With the keyboard, you can resize more than one selected item at the same time.

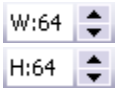
As you move or resize an item with the mouse, a little ToolTip displays either the item's position (relative to its parent Container) or its dimensions – depending on the operation in progress.

You can constrain a moving or resizing operation to a single axis (horizontal or vertical) by pressing and holding the Ctrl key once you have started the moving/sizing operation. For example, if you want to move an item horizontally, but you want to make sure it does not change its vertical position, you can start dragging it, press Ctrl, and keep dragging it approximately along the horizontal axis.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

You also have some toolbar controls that allow you to perform a few operations related to item moving and sizing.
When a single item is selected, the only relevant thing you can do to it is explicitly set its size in pixels.



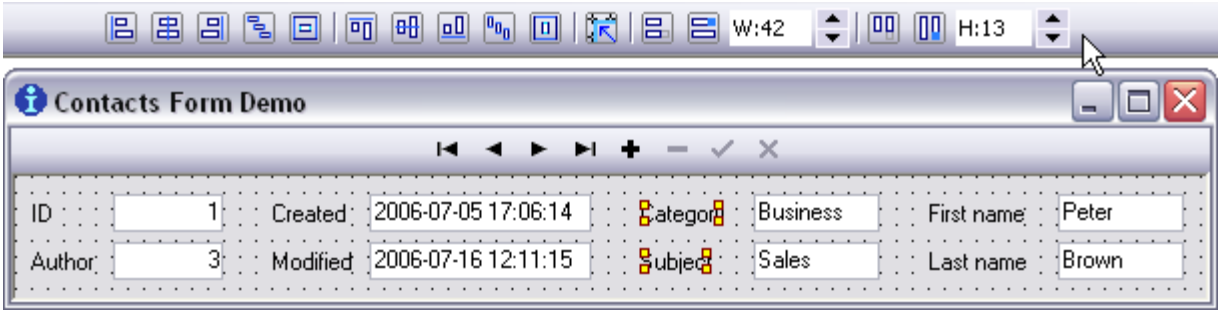


Set Width SpinEdit sets the item's width

Set Height is similarly responsible for the height

As always, you can change the values using the SpinEdit's little arrow buttons, by clicking the Up/Down keys on the keyboard, or by typing the desired value directly.

Whenever there are 2 or more selected items, quite a few more commands become relevant and thus available.



Here are the horizontally-moving ones:

Align Left Sides will move all selected items to line-up with the leftmost of them

Align Centers (H) will line-up their midpoints

Align Right Sides lines-up the right sides

Space Equally (H) spreads them in the horizontal direction

Center in Parent (H) centers them in whatever happens to be their container (more on these later)

The horizontally-sizing ones are self-explanatory:

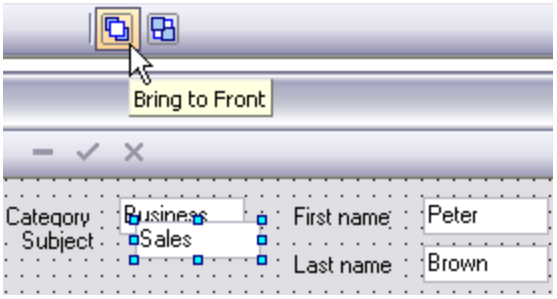
Shrink Width to Narrowest

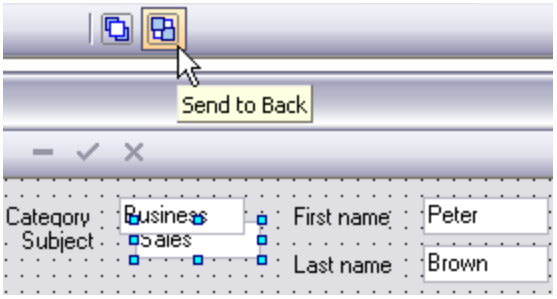
Expand Width to Widest

There are also the vertical counterparts of all of the above that perform similar actions in the up/down direction, so we won't bother listing them.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Z-Order. You may occasionally need to adjust what is known as the Z-Order of items. This simply means which item is "on top" of which. The "Z" refers to the imaginary Z axis sticking out of the InfoView, perpendicularly to its surface. And you thought Designer was a 2D application! Adjusting Z-Order becomes important when you have overlapping items. We haven't seen anything like that yet, but let's get it out of our way. There are two buttons that let you adjust the Z-Order:





Bring to Front places the selected items on top of the rest of them

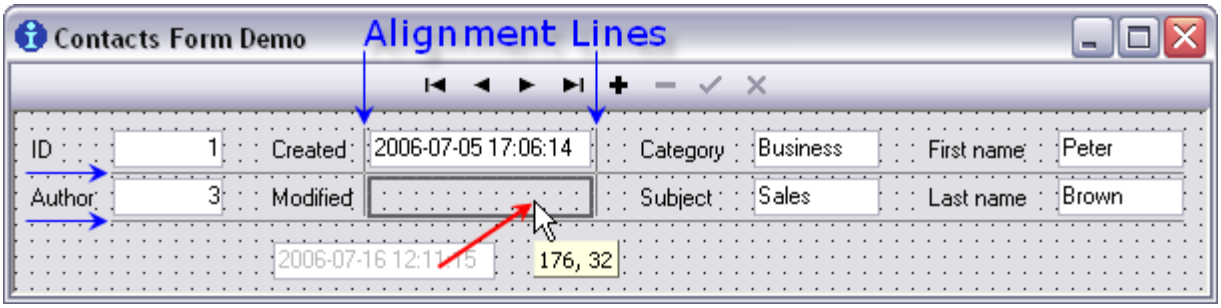
Send to Back does the opposite. This is what we'll use at a later time to make sure a background picture appears underneath all other items instead of covering them, for example.

There's one peculiarity about how items are stacked on top of one another that you should be aware of. Items basically fall into two groups. For lack of a better term, we'll call them Lightweight items and Heavyweight items. The Lightweight items are of a cosmetic or static nature and include: Label, Rectangle, and Image (the static kind used for esthetic purposes, not the field-based one). All the other items are Heavyweights. Such items can be regarded as functional or dynamic. The peculiarity you should be aware of is that all Heavyweights are always "floating" above all Lightweight items in the Z-Order. You can re-order Labels and Rectangles, for example, but there's absolutely no way to make a Label appear above, e.g., a Panel. Inside a Panel – yes, no problem (because now the Label is parented to the Panel, so it is their parent-child relationship that takes over and Z-Order plays no role), but if the Panel and a Label are siblings (both parented to a common Container), the Label will always be underneath.

This can sometimes be a bit of a limitation. What if you wanted, for example, to display a Label or warning or a static Image on top of, say, a Grid item? You can't do it. At least not without resorting to a simple trick. Place a little Panel above the Grid (or whatever Heavyweight you are trying to "cover") and place the Label or Image inside that Panel. The Panel is also a Heavyweight, so it can "float" above whatever you want. Whatever is in it is no longer "floating" above the Grid item, but merely "sits quietly" in its Panel Container. You may not find yourself performing such Z-Order "stunts" very often, but at least you know that there's a way if you need it.

Alignment Lines. There's a very nice visual aid to help you align and resize items relative to one another, and you may have already seen it in action if you've been playing along. Alignment Lines are the lines that appear automatically as soon as you attempt to move or resize item(s) with the mouse. As soon as relevant edges of the controls line-up, the corresponding line is drawn to indicate this joyful event.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Gridlines. Another feature aimed at helping you create well-aligned InfoViews is Gridlines.

You can display or hide these dotted lines that form a visual "grid" over the InfoView using the Toggle Gridlines button.

8 The Gridline Step SpinEdit controls the distance between the lines of the "grid". The default is 8, but you can set it to whatever suits you.

Independently of whether the Gridlines are visible, you can make moving and sizing operations respect the Gridlines. When Auto-snap to Gridlines toggle button is in the depressed state, moving and sizing an item will make it "jump" in the desired direction by whatever number of pixels Gridline Step happens to equal (instead of moving smoothly 1 pixel at a time).
You can temporarily suppress the Auto-snapping (if it's in effect) by performing the moving or sizing with the Alt key pressed. Alt-moving or Alt-sizing are therefore always smooth as if Auto-snap were Off.

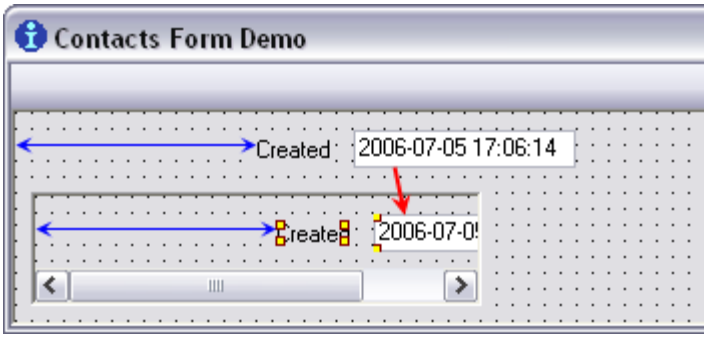
If you ever want to align some existing items to Gridlines, the Snap to Grid button accomplishes just that. This button looks similar to the Auto-snap to Gridlines button, but this one is a bit smaller and enclosed in a rectangle.

Custom Gridlines, when depressed, draws the Gridlines using a crosshair pattern, which may be more visible on some forms. Otherwise, Gridlines are drawn as simple dots.

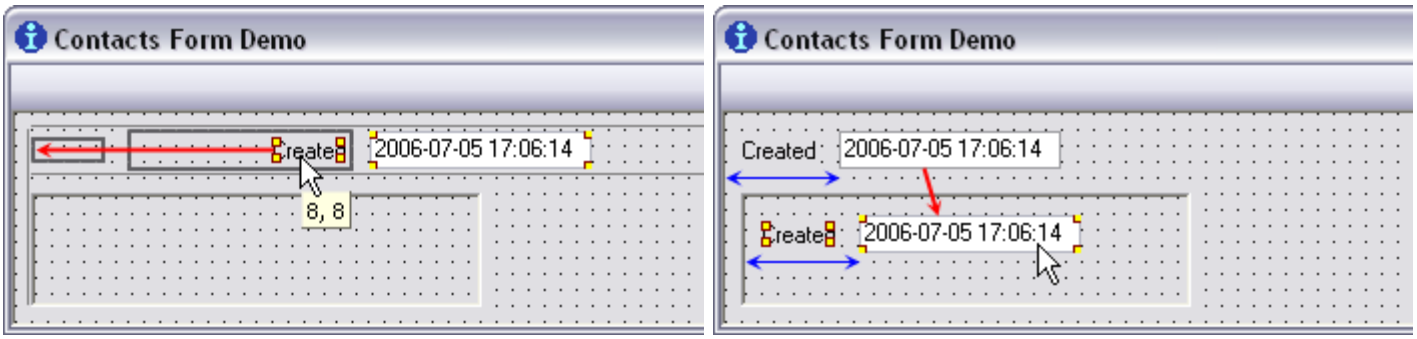
Cut-and-Paste. You can use all the usual editing operations on the selected items: Copy-and-Paste, Cut-and-Paste, and Delete .

When using Cut-and-Paste between different Containers (which is what you'd normally do to "move" items from one Container to another), Alventis maintains the position of the items relative to their parent Container. This may be a bit inconvenient when pasting an item that had relatively large X or Y offsets into a smaller Container: the item gets pasted and positioned with its old (large) offsets, so it may end up beyond the visible area of the new Container.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



You can scroll to it and move it where desired, but there's a neat trick that makes this unnecessary. Simply move the item(s) to reduce their offset within their old parent Container before performing the Cut operation. The cut item(s) will now maintain smaller offsets when pasted, so they will fit in the new Container.



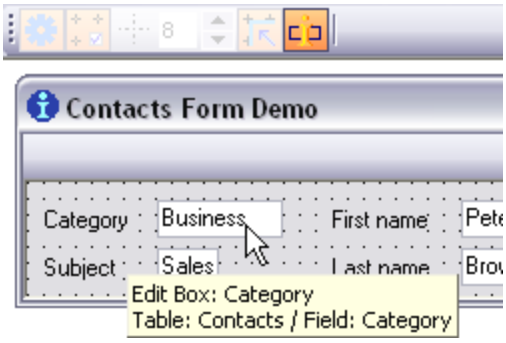
Warning. Please note that Alventis has problems performing a *Copy*-and-Paste operation on a Grid within the same InfoView form. Pasting Grids into InfoViews that already have other Grids may also cause problems. This should not be something you'd want to do anyway, so please avoid using such Clipboard operations with Grids. *Cut*-and-Paste within the same InfoView should work fine though.

Design Mode. Normally, you'd want your InfoViews to be in Design Mode where you can create items, adjust them, and so on. After all, that's the whole point of Alventis Designer. Even in Design Mode the InfoView is shown almost exactly as it would appear in Alventis. On occasion, you may want to "take it for a spin" though. One reason could be: you want to adjust the data in the fields to make sure the corresponding Controls are large enough. Another: to verify that the Tab order of all Controls is correct.

You can therefore temporarily disable Design Mode using the Design Mode toggle button . When in the Off state, it places the InfoView in Preview Mode that's pretty close to how the InfoView would behave in Alventis. You can edit the current record's data or create a new record. This is precisely what gives you the ability to make sure you have some sensible data displayed in the Controls – so that you could size them appropriately to their intended contents. Once satisfied with your data edits, you can return to Design Mode and make whatever adjustments may be necessary. What you cannot do in Preview Mode is post a record. Whatever you do with the data is temporary and for your convenience only. If you want to try some real editing, you must open the InfoView "for real" in Alventis.

Item Hints are item-specific ToolTips that display some basic information about the item you pass the mouse over. Enable/disable them using the Design Item Hints toggle button .

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



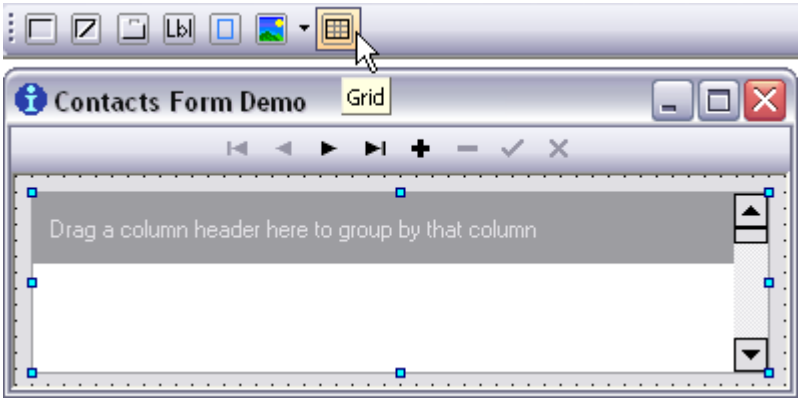
Note that this is essentially the only way for you to "identify" an item on an InfoView, i.e., get an answer to a question "Dear Panel, what is your name?" – which is something you may occasionally want to do if there are so many Panels you get confused as to which is which.

3.6 InfoView Grids

We have already learned how to create fields and place the corresponding Controls on an InfoView. There's an important alternative to what we've seen so far, and we are going to examine it here.

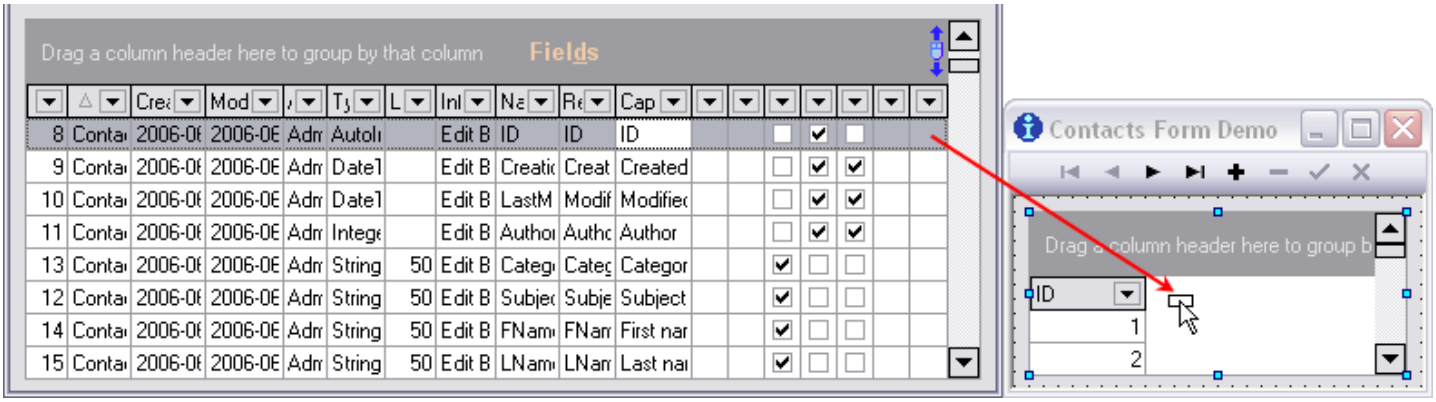
You may have gotten used to grids by now: both Alventis and Designer use them all over the place. Designer allows you to create such grids on your own InfoView forms. The resulting grids can represent the entire table the InfoView is based on or any part thereof. Let's see how we would go about creating one.

An **InfoView grid** (any grid for that matter) does not represent a single field, but rather a collection of fields, possibly even all fields from a table. This is why you won't find the "Grid Control" among the choices you have for any given field. Instead, to create a grid, we simply invoke the Grid command. Clicking this button creates an empty grid and places it in the InfoView.

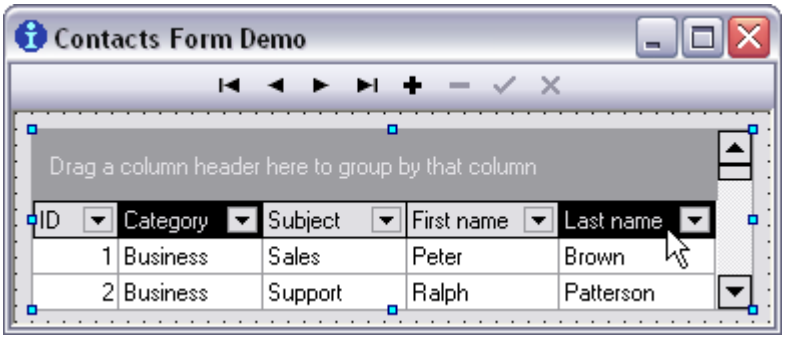


Now that we have our grid in place, the rest is easy. To add a column "implementing" a field of the table... can you guess?... just drag the field and drop it onto the grid. Or double-click the field. Or drag the table itself from the Tables grid. Basically, the grid acts as a "recipient" of fields – exactly in the same way the whole InfoView does.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Editing of grids as far as moving and sizing is concerned is somewhat different from how it's done for regular items, but it should be mostly quite intuitive. The thing about the grid is: the whole grid is an item – just like an Edit Box or a Label. You can select the grid as a whole, and move or resize it exactly as we've done for other types of items. What makes the grid special is that it contains other stuff within it, the most important being obviously the columns. Turns out we have virtually the same amount of control over the columns as we have over other items. Resizing of columns works just as it does in all "regular" grids: just drag the edge between the column headers with the mouse. You can select individual columns by clicking them. Since the Shift key applies to selecting/deselecting multiple items at the "real" item level (that of the whole grid), we can't use it to select multiple columns though (don't worry if you didn't follow this sentence). You can select multiple columns, "adding" to an existing selecting by Alt-clicking columns. You de-select all columns by clicking somewhere outside all columns in the grid. The Group by Box at its top may be a good place, for example.



You can delete selected columns using the Delete command. We'll see later what else you can do with them. One thing you unfortunately cannot do is re-order the columns. In Design Mode, that is. Go temporarily into Preview Mode (by clicking the Design Mode toggle button) – and the grid starts to act as any other grid, allowing you to drag-reorder columns, use grouping and sorting. While you're at it, you might as well create some Summaries too. This works exactly as it does in Alventis: enable the Summary rows you want using the Grid Group Summary or Grid Group Summaries buttons, then double-click the rows to cycle through available Summaries. Once satisfied with your adjustments, you can return to Design Mode and go back to "business as usual".

Just as with ordinary "standalone" Controls, the type of column that you create for a field depends on what Control you have selected for it in the Fields grid. A field with a Calculator Edit Control will, for example, create either a standalone Calculator Edit Box or a Calculator column.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

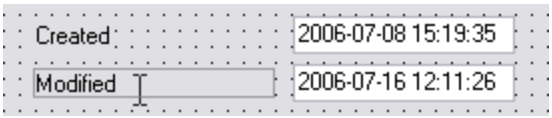
3.7 InfoView Items Overview

We have already encountered quite a few interesting items one can place on an InfoView. So far, we've seen:

- Controls, which includes a whole palette of items
- Labels that typically accompany Controls
- Grids which we have examined in detail in the previous chapter

We will now take a closer look at some of the field Controls and see what, if anything, we can edit about them.

Label. This is perhaps the simplest of all items, but there's a trick even to this one. You can edit the Label's Caption simply by double-clicking it.

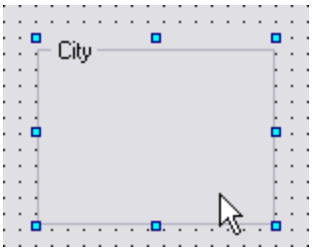


The Caption "lets you in": you can adjust the text in the Text Box it has temporarily become. Just click somewhere outside the Label or hit Ctrl-Enter when you're done – and the Label is updated.

Checkbox. While not a Label at all, it sort of looks like one, doesn't it? Editing its Caption therefore works exactly like editing a Label: just double-click it to edit.

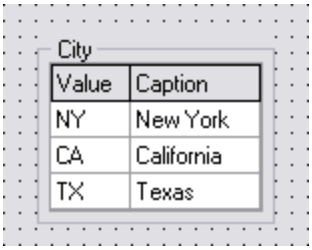
Grid Column Caption. It's not a separate Item per se, but it works like a Label: double-click to edit.

Radio Button Group. Implementing a field using this control creates an empty Radio Group with no Buttons inside.

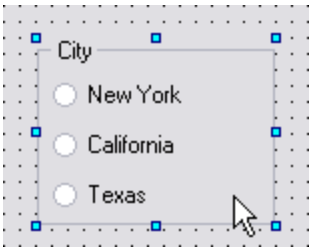


The way you create the buttons that will correspond to whatever choices you want this Radio Group to present, is by getting "into" the Radio Group. You do this by double-clicking it – right in the middle. This opens a little grid of Value/Caption pairs. The Value corresponds to the actual "real" value this Radio Group will store or retrieve from its underlying field, e.g., "NY"/"CA"/"TX" and so on. The Caption is the caption of the corresponding Radio Buttons, which in this case would be "New York"/"California"/"Texas", etc. Using this example, we'd fill the little grid to make it look like this:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Once satisfied with the setup, you can close the mini-grid simply by clicking anywhere outside the Radio Group. This creates the Radio Buttons in the order in which you have listed the corresponding Value/Caption pairs.



Note that you can't re-order the pairs at will, so you have to be very careful and organized as far as the order in which you enter them: start with the last Button, then insert the other ones. If you do find yourself in need of re-ordering the Radio Buttons, you can sort their list by either the Value or Caption column – in ascending or descending order. If neither order produces the right results, here's a little tedious trick: edit the Captions so that their first character places them in the desired order (e.g., using the example above, you'd change them to "1New York"/"2California"/"3Texas" or similar). Sort by the Caption column and accept your changes. The order is now correct, but the Captions themselves aren't. Re-open the grid, and "fix" the Captions by taking out that first character. Re-apply the changes. The already-correct order has not changed, so now you have your correct Captions in correct order.

If you want to change the Radio Group's Caption, editing it works exactly as if it were a Label (see above): double-click, edit, click elsewhere or hit Ctrl-Enter. This is the only reason we told you earlier to double-click the Radio Group close to its center: in fact, you must double-click it anywhere outside the Caption area at the very top.

Immediate Lookup Combo. Since it performs essentially the same function as a Radio Group, it too requires a set of Value/Caption pairs, which are edited the same way: double-click the Combo Box.

Lookup Combo Box. We mentioned it here only for the sake of completeness, but we really don't want to talk about it until the chapter on Relational Tables. So, don't double-click it!

You have already seen a few other Controls, e.g., Calculator Edit Box. We are not listing them here simply because there's nothing to be edited about them – at least not in a way similar to those we did list above.

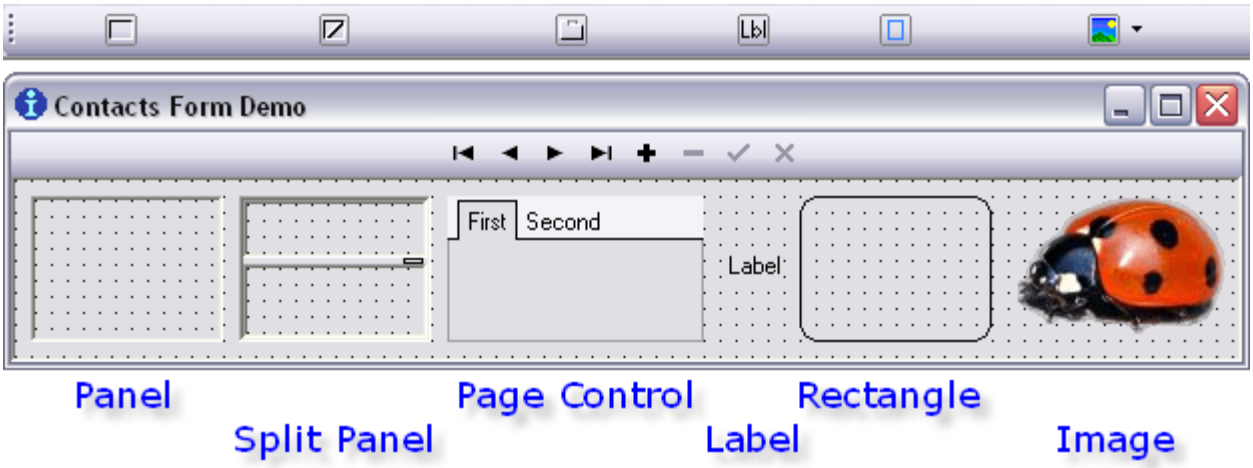
Symbol Picker. This is not directly related to item editing, but you will most likely use it to enter fancy symbols into Label Captions and the like, so we'll mention it here. Basically, the Symbol Picker

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

works exactly as it does in Alventis. The only difference is that in Alventis, entering a symbol belonging to a particular Character Set (e.g., Greek) would affect the Charset of the entire item (Text Box, etc.) into which you are entering it. In Designer, it doesn't. If you need to set the Charset of an item, you have to do so explicitly.

As you can see, quite a bit of useful stuff can be edited directly, "in-place", simply by double-clicking the corresponding item.

Let's see what other items we can create in Designer.



Panel. A Panel is a very important item. It allows you to place other items (including other Panels) onto it. Whatever is placed on a Panel will then "follow it around", i.e., move where it moves, gets resized with it, etc. We will talk much more about Panels and their usefulness in the chapter on Layout.

Split Panel. A Panel with a Splitter that separates it into two sub-Panels. Again, we'll leave it aside until we talk about Layout.

Page Control. This is the familiar multi-page control comprised of multiple Tab Sheets. A very different kind of item, and definitely the kind we want to set aside until the Layout chapter.

Label. We have already seen many of those: they get automatically created to accompany most Controls that implement fields. You can create your own "independent" Labels too.

Rectangle. This is just a rectangle, no more, no less. Useful for surrounding logically-related Controls, for example. You can make its border or its inside area whatever color you want. To make them transparent, set the corresponding color to "None".

Image. This is just a static picture box: an item into which you can load a graphic file of your choice. It may be used for decorating your InfoView or whatever other esthetic purpose it may serve. Do not confuse this static Image item with a "real" Image Control based on a Graphic field! The former gets assigned some image for purely cosmetic purposes and never changes. The latter displays pictures stored in the Database table and allows you to save/load them. If an existing Image item is selected,

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

invoking this command will assign a new graphic file to the existing item. Otherwise, a new Image item will be created with the picture of your choice. You can select one using the standard Open dialog you get if you click the button. The drop-down menu attached to it lists the most recently used pictures, so you can select frequently used ones from it directly.

Layout Template. This is not an item in any sense of the word, but inserting it is quite similar to inserting ordinary Panels described above. A Layout Template is simply the contents of any InfoView form that has no data-aware items and hence has not been "attached" to any table. This is really simpler than it sounds. Imagine you have devised a certain layout consisting of the above-mentioned non-data-aware items, such as Page Controls, Panels, Split Panels, Shapes, and perhaps even Labels. You may have arranged them in a certain magnificently complex structure with all sorts of clever Alignments and Anchors (that you will learn about in the next chapter). To reuse your work, you could certainly use Copy-and-Paste, but there's a better way. Simply click the Insert Layout Template button. This will open a list of InfoView forms that have not been attached to any data, which should therefore include the InfoView with that clever layout you want to reuse. Select the desired InfoView from the list – and its entire contents is inserted into the currently selected Container. Like we said, this works very much like inserting a Panel or similar – except that in this case you are inserting whatever was in the selected "template" InfoView.
Creating a Layout Template is therefore simple: just create an InfoView but do not place any data-aware items on it (i.e., do not drag any fields into it). You can create and save it attached to any Table you wish. You may find it convenient to create all Layout Templates under some particular table – just to keep them together. This is up to you though since Designer will find all suitable InfoViews regardless of where you put them. You will find the several default ones under the Car Manufacturers table (simply because that table has the fewest "real" InfoViews).

The above represents absolutely all items you may place on an InfoView form. The list is not particularly long, so you should be able to quickly and easily master every available item.

We have intentionally avoided quite a few attributes of all these items. Some will be treated in the next chapter on Layout. Others will get what they deserve in a separate chapter dedicated to Item Formatting.

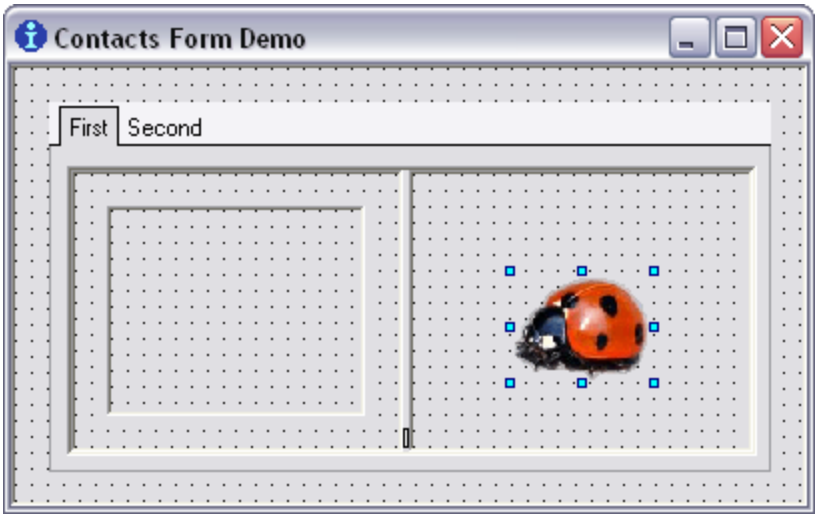
3.8 Layout

Alventis Designer gives you all the tools you may need to produce InfoViews that are not merely functional, but also exceptionally well laid out. In this chapter, we'll learn what these tools are and how to use them.

In Alventis, almost all essential forms are resizable. InfoViews are no exception. It is therefore very important to understand what happens to the form's contents when it gets resized. But first, we'll have to learn about Containers.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

A **Container** is simply an item that can contain or hold other items within it. The InfoView form itself is one such Container. The other ones are: a Panel, a Split Panel, and a Page Control. Nothing else "qualifies". An Edit Box can't hold a Checkbox inside it. An Image item may initially seem like a Container, but in reality it's not: you can place items on top of the Image, but not in it. Even a grid is not a Container. While it is true that it can hold columns, these are not items in the general sense, but rather the grid's own sub-elements (even if we occasionally refer to them as "items"). The ultimate test is quite simple: moving a Container moves the items it contains. You can move an Image item all you like, but whatever other items happen to be on top of it won't move with it, so it "fails" the test.

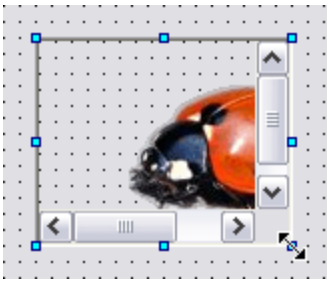


We will be saying about a Container that it is *parent* to the items it contains, which we will refer to as its *children*.

Why all this fuss about Containers? Well, for starters, because all items exist in their parent item and whatever layout-related behavior they may have is relative to their parent.

Whenever you are resizing a Container (you personally will be resizing the InfoView, but it may be resizing some child Container within it), there are two issues to consider:
A. How the Container is reacting to its contents
B. How the Container's child items are reacting

A. The first one is rather simple. The Container only reacts to child items within it that do not move when the Container is resized. And the way is reacts is: whenever it is shrunk so much that at least one non-moving child item no longer fits, the Container displays appropriate scrollbars thus extending its "virtual area".



It may sound mighty complicated but you have undoubtedly witnessed it a hundred times: make a form too small and suddenly it's got scrollbars, so you can scroll around to still get to all these (child) items.

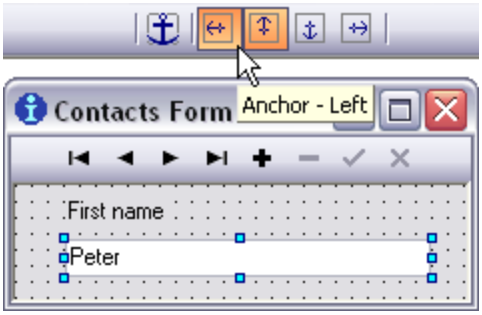
B. The possible reactions of the Container's child items to their parent's resizing vary between very simple and rather involved:

- Do nothing. Nothing gets simpler than that. The Container resizes – the child item "stays put" and doesn't budge. Nothing interesting here – except that it is precisely such "dumb" items that make the Container react by displaying its scrollbars when necessary.
- Move and/or resize itself according to the child's Anchors property. We'll see what this means in a bit.
- Resize itself proportionally to the parent Container. This sound like the most natural thing to do, but

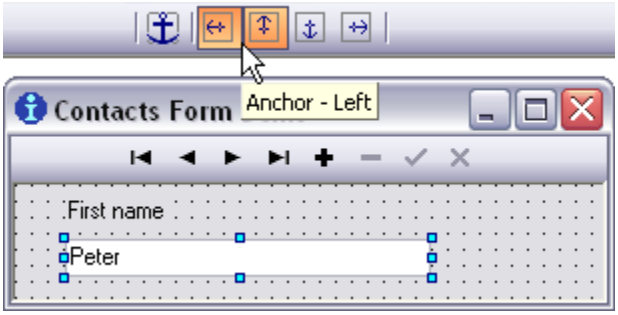
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

amazingly, only Panels and Split Panels can do this. We will learn how exactly this works shortly.

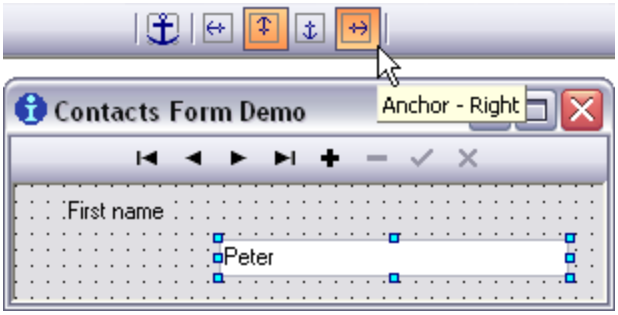
Anchors are the cornerstone of Layout, so we'll have to thoroughly familiarize ourselves with what they are and how they work. It's actually quite simple.



Every item has 4 Anchors: Left, Right, Top, and Bottom – one on each side. Each of the Anchors is like an On/Off toggle switch, so an Anchor can be On or Off. They are all pretty much independent from one another, and each of them means the following thing: when On, maintain the distance from the item to its parent on the corresponding side (when Off, don't). Let's see how it works for, say, an Edit Box directly parented to the InfoView (i.e., the Edit Box is child of the InfoView as all our items have been so far). We'll see what will happen to it depending on how we set up its Left and Right Anchors.
By default, the Edit Box (almost all items, in fact) has its Left and Top Anchors On and the Right and Bottom Off.

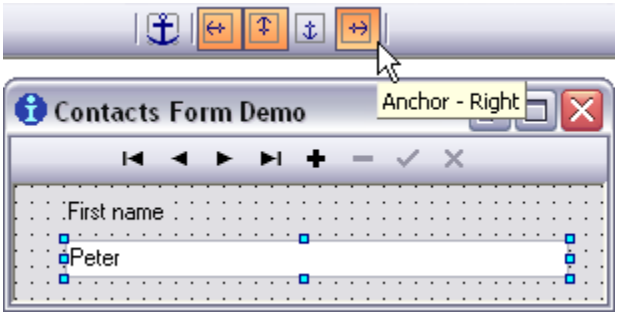


We promised to only play with the Left and Right ones, so we'll leave the Top/Bottom ones alone. So, initially, the Left Anchor is On, Right one Off. We all know what such Edit Boxes do when their parent resizes: nothing. This makes sense: the Left Anchor tells the Edit Box to maintain its distance on the left from its parent's left edge – and that's what it does. It's not its fault that it looks like it's not doing anything.



It's also true that doing what it's doing (staying in place) doesn't take much effort, so let's make its life a little more complicated. Let's turn Left Anchor Off and Right Anchor On. But we haven't told you how to do this, have we? The Anchor - Left and its 3 sibling toggle buttons do this. Now, when we resize the InfoView, our Edit Box moves "with" the right edge of the form. This is exactly what we have told it to do: maintain its distance on the right.

On to the most interesting test now: turn both the Left and Right Anchors On.



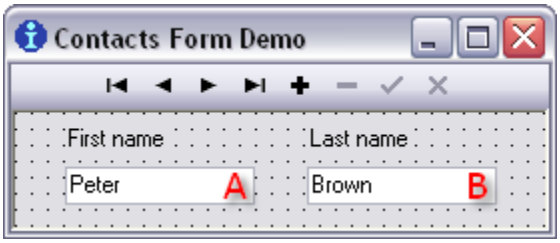
As is to be expected, the Edit Box now maintains both "margins" constant, and resizes itself as necessary.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

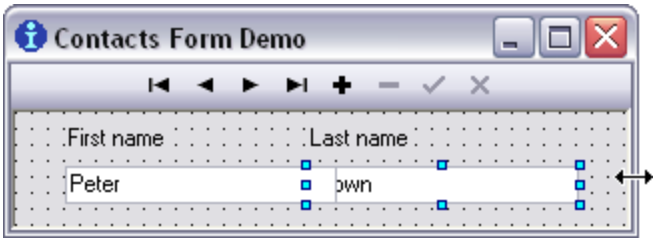
This essentially concludes our tests, but there remains one setup we haven't tried, so let's do it for the sake of completeness. Turn both the Left and Right Anchors Off. Did you guess what the Edit Box would do? It now seems to make a great effort to not maintain its distance on either side by moving proportionally to its parent's width. It looks rather neat but the only meaningful use we have ever found for this behavior is centering an item within its parent, which is not something you are likely to use very often.

Top/Bottom Anchors work similarly to their Left/Right counterparts – but in the vertical direction. Now that you have seen Anchors in action, you can hopefully appreciate their importance and power. You are not likely to ever create a nicely resizable InfoView without adjusting at least some Anchors.

Before we move on, we'd like to point out to you one common source of confusion.



Given the item layout depicted in the accompanying screenshot, how do we set the Anchors of the two Edit Boxes so that they maintain the 3 gaps between themselves and the edges of the form?

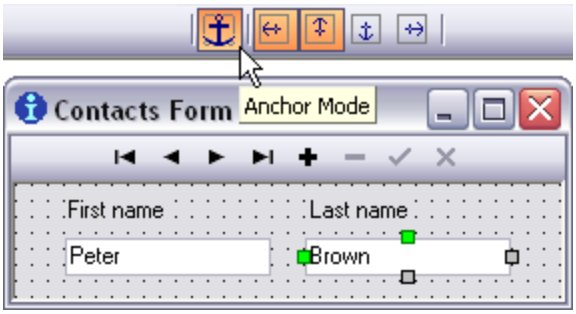


The frequent initial reaction may be to set both Left and Right Anchors to On, and that – for both Edit Boxes. Alas, the results are not what we could hope for. You can try it yourself, but the cause is easy to understand even without experimenting: Edit Box A will maintain its distance to its parent on the left (great!) and... on the right – and this is precisely the problem. On the right, it will be maintaining its distance to its *parent* (the form) and not to Edit Box B. The latter will have the same exact problem on its left side. As a result, both Edit Boxes will be resizing themselves, quite oblivious of one another, overlapping, etc.

The correct solution to this problem... there is none. At least not without resorting to using a few more "intermediate" Containers, which is what we are going to examine shortly.

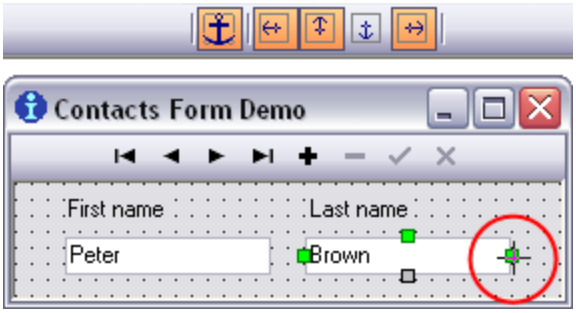
Anchor Mode. Properly setting up the items' Anchors is so important that Designer offers you an innovative method of inspecting and setting them. The basic idea is this: you temporarily put the InfoView into a special Anchor Mode that helps you adjust Anchors, you perform whatever adjustments may be necessary, and return to the normal Design Mode by deactivating the Anchor Mode.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Anchor Mode toggle button , as the name suggests, is responsible for turning this special mode On or Off. Once in the Anchor Mode, everything behaves a little differently. You can select only one item at a time. You cannot move or resize items. The only thing you can do is what this mode is all about: inspect and adjust Anchors. And this task is remarkably easy. The selected item displays the Anchors in the middle of its edges.

The Anchors that are On, are shown as green squares, those that are Off – as gray ones. You can toggle an Anchor by clicking it with the mouse.



You therefore have a much more direct visual feedback – right at the spot where the Anchor is "attached" to the item. The limitation of this mode is of course that you can only adjust Anchors for one item at a time. If you have a whole array of items you want to set Anchors for the same way at the same time, you may be better served by the "regular" Anchor buttons. Make sure Anchor Mode is Off, select all the items you want to affect, and toggle the Anchors using the buttons. You won't have any visual feedback directly on the affected items, but at least you don't have to iterate through them one-by-one to set them up.

Proportional Panels. A Panel is a Container item. As such, it can hold other items inside. Any items. A Panel can have a border – and by default it does have a thin 3D sunken one. We'll learn to adjust its appearance later. What interests us here is the Panel's layout-related behavior.

Remember the 3rd possible kind of reaction of a Container's child item to its parent's resizing:
- Resize itself proportionally to the parent Container.

Panels possess the unique ability to do precisely that.

By default, Panels behave (layout-wise) just like other items. They too have Anchors, and they act in accordance with how they are set up. But one flick of a switch – and all this changes. The "switch" is appropriately called Proportionality, and there are two independent switches: one for the horizontal and one for the vertical direction .

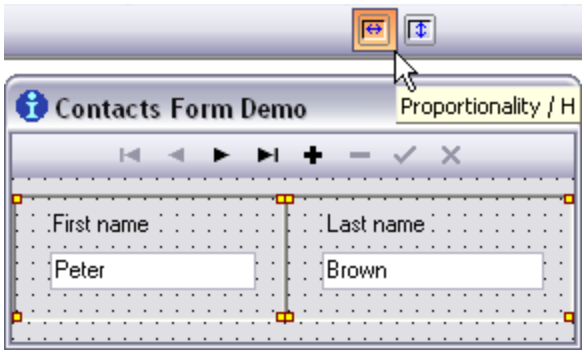
Proportionality / H, when enabled, makes the Panel resize itself proportionally to its parent Container (sorry to keep repeating ourselves) in the horizontal direction.
We should note that Anchors and Proportionality are mutually-exclusive: you can only use one or the other, but never both (which wouldn't make much sense anyway).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

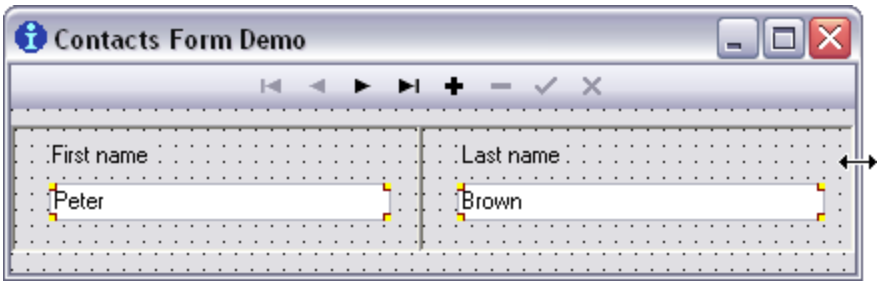
Let's go back to our last problem with two side-by-side Edit Boxes and see if Panels can help.

We will now place two Panels side-by-side, completely "filling" all available space (horizontally).

We'll put Edit Box A in the left Panel and Edit Box B – in the right one:



With that done, we will make both Panels Proportional in the H direction by selecting both and making sure Proportionality / H button is depressed. Almost there. We now set both Left and Right Anchors to On for both Edit Boxes (yes, just as we have unsuccessfully tried earlier). Done!



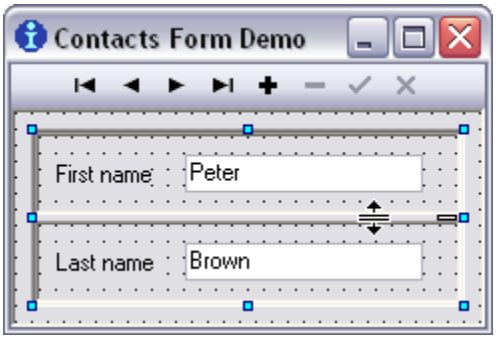
Why does it work now? Let's see... Each Panel maintains its size and position proportional to that of its parent (independently from one another: they know nothing about each other and the fact that they are touching edges is purely incidental). The left Panel will therefore always occupy the left half of the form, and the right one – the right half. Each Edit Box is anchored to its parent, which is now its respective Panel. So, each Edit Box will be maintaining its distance to both edges of the Panel, which corresponds to the left edge of the form (because that's where the left edge of the left Panel happens to be) and its middle (because the right edge of the left Panel makes an effort to stay there).
You can also think of it this way if you prefer: by using the Panels the way we did, we have split the whole problem in two. Each panel behaves independently from the other, so all we care about now is that one Edit Box behaves properly in one Panel. Which is real easy. Now repeat that twice – for both Panels, and we're done.

To generalize this: properly set up Proportional Panels split their Container in independent pieces, so you get a much better chance to make each individual piece work the way you want, which ends up making the whole Container work.

Some layout tasks may require just a few Proportional Panels on the form. Other, more sophisticated ones, may benefit from multi-level Panel-in-a-Panel-in-a-Panel setups. No matter how complex it gets though, the basic principle is always the same, and we've just seen it "at work".

Split Panels. A Split Panel is a Panel with a Splitter that divides it into two Sub-Panels.
You can adjust the position of the Splitter with the mouse simply by dragging it wherever you want. Technically, doing so resizes the two Sub-Panels according to the Splitter's position.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

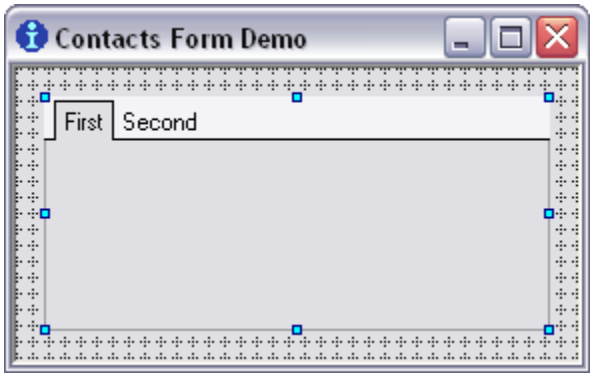


While it may look for a moment like a very different item, the fact of the matter is: a Split Panel is just a regular Panel (the kind we have examined earlier) that holds two more such Panels inside, with a Splitter in-between. That's all there really is to it, so there's very little new stuff to learn. It mostly works and reacts just the way ordinary Panels would. You can set the Split Panel's Anchors or Proportionality. The Sub-Panels, while true Panels in all respects, offer you fewer layout adjustment possibilities simply because they are already properly laid-out for you within their parent Split Panel. This is why you can't even resize them. You can still set Proportionality for the "top/left" Sub-Panel though. Resizing the entire Split Panel will make that Sub-Panel resize itself proportionally, which will have the effect of the Splitter automatically maintaining the split ratio, e.g., one third on the top, two thirds at the bottom.

The Splitter is a RotoSplitter of course, so you can rotate it 90 degrees at any moment by Shift-double-clicking it or by clicking the little button at its right/bottom end.

Selecting Sub-Panels is no different than selecting any other item: just click them. Not so with the Split Panel itself. Since it is completely covered by the Sub-Panels, there's no place to click. One solution is to use the keyboard. The Tab and/or Arrow keys could do the job. An even better way is to select one of the Sub-Panels and hit Escape. As you might recall, Escape always attempts to select the parent Container of the currently selected item, so it will select the Split Panel. That's all very nice if all we want is to *select* the Split Panel, but it doesn't help much if we want to move it somewhere by dragging it with the mouse: the moment we try to click it, one of the Sub-Panels get selected. The solution is to Alt-click or Alt-drag it. Alt-clicking either Sub-Panel selects the Split Panel. Likewise, Alt-dragging will drag the entire Split Panel.

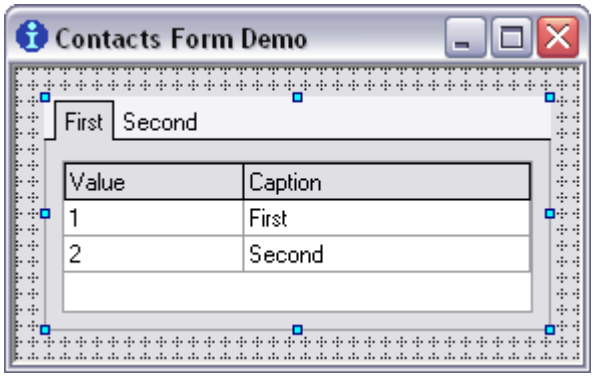
Page Control. The last of the Containers.



You are probably familiar with these ubiquitous and frequently over-used user interface items. A Page Control can have one or more Tab Sheets. The little tabs on the top allow one to select the active Tab Sheet that will then reveal its contents to the world. This can be a neat way of organizing a busy form that has too many items to fit in a single screen. Quite frequently, Panels and/or Split Panels prove sufficient. Even if your monitor's resolution is not very high, and you can only see small Panels with items crammed in them, you may be able to "zoom-in" to a specific Panel using PanelZoom. Somebody else may have a high-resolution monitor though, and he/she will benefit from seeing all form's items at the same time, side-by-side, which is usually preferable. For situations when such a Panel-based solution is not what you're looking for, the Page Control provides an easy alternative.
Placing a Page Control on an InfoView creates a default Page Control with two Tab Sheets. Manipulation of Tab Sheets is very similar to how you work with, e.g., a Radio Group: double-click the Page Control to display a mini-grid of Tab Sheets.

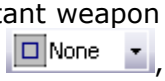
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The Caption column lists the Tab Sheets and allows you to adjust their tab's Captions. The Values column, on the other hand, is not very meaningful. Its only purpose it to make it easier for you to re-order Tab Sheets. You can enter numbers or letters into it and sort the grid by that column. The Tab Sheets will end up in the same order they appear in the list, so whichever way (and by whichever column) you sort, this will determine the Tab Sheets order in the Page Control. The mini-grid is obviously where you can create new Tab Sheets by inserting a record in the list or delete them by deleting the corresponding record.



As far as selection goes, there are no surprises here. You can select the entire Page Control by clicking on the row of tabs at the top. You can activate individual Tab Sheets by simply clicking on the tabs – exactly as you would do in any Page Control. You can select a Tab Sheet by clicking it.

Tab Sheets always occupy the full area of the Page Control allotted to them, so you can't resize or move them. The Page Control itself can be moved/sized in the usual ways and you can set its Anchors.

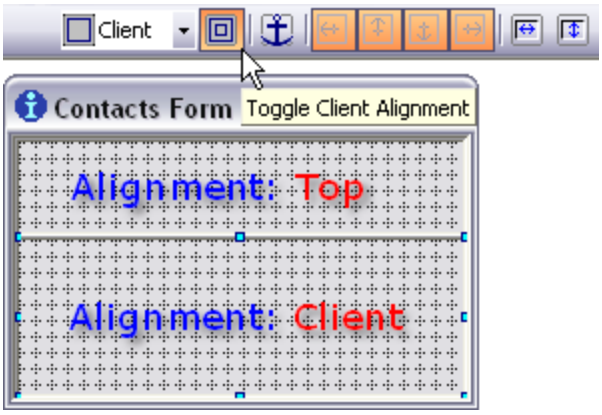
Alignment. There's one more important weapon in our Layout arsenal, and it's called Alignment. The default setting for all items is "None" 

, i.e., no particular Alignment. This means the item can reside wherever we place it within its parent. If we take a Panel and use it for our examples, and set its Alignment to Top, the following happens: the Panel "jumps" to the top of its parent Container (we can assume it's the InfoView form for the sake of simplicity), and gets "glued" to its top, left, and right edges. Only the bottom edge is "free", so we can resize the Panel by dragging the bottom grab handle up/down.
Alignment Left, Right, and Bottom work in a similar fashion: the Panel "sticks" to the specified side of its Container.

The only remaining type of Alignment is Client. You can also toggle the Alignment between None and Client using the Toggle Client Alignment button . What it does is make the item (Panel or whatever) fill the entire client area (i.e., available/usable area) of its parent Container. The so-aligned item "sticks" to all sides of its parent. What good is that? Let's take a look at some examples.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

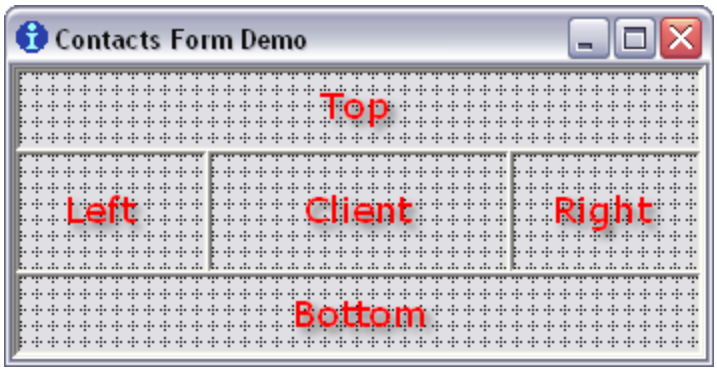
First, let's place two Panels on an InfoView. For the first panel, let's set its Alignment to Top and ensure the Panel occupies roughly the top third of the form. For the second Panel, we'll set its Alignment to Client. The second Panel now occupies the entire available area, i.e., whatever area was not taken by the first Panel. Resizing the first (Top-aligned) panel automatically adjusts the size of the Client-aligned second Panel. Resizing the entire InfoView form doesn't affect the height of the Top-aligned first panel, but the second one always fills whatever space is left, so it gets automatically resized. This is a great way to very quickly divide a form (any Container really) in portions that will neatly maintain their overall relative layout.



You can put more than two aligned Panels next to each other.

Here's an example of just one possible layout.

One common thread that is likely to be found in all such layouts is the single Client-aligned Panel that always "completes the picture". You are under no obligation to have one, but you'll discover that your layout will be mighty strange without it.



Here's another, even simpler, example of when you might want to use Alignment. A Page Control by itself does not "know" how to behave proportionally. If you want to have a layout where a Page Control resizes itself proportionally to its Container, e.g., always occupying the left half of it, you have to use some persuasion. What you'd do is: place a Panel on the Container and make it Proportional in the right direction(s). Place the Page Control into the Panel. All we need now is to make the Page Control fill the entire area of the Panel. Sure, we could carefully resize it and set all of its Anchors to On (and yes, it would work fine), but there's a much easier way: just set the Page Control's Alignment to Client – and it does precisely what we want: covers the entire Panel. The Panel now behaves proportionally, and the Page Control within it simply keeps up with it by resizing itself to whatever the Panel decides to size itself.

Many items support the Alignment property, but not all of them would benefit from it. Items that look and essentially behave like a rectangle of no pre-set size would usually be good/sensible candidates for setting their Alignment: Panel, Split Panel, Page Control, Grid, Image, Rectangle, Memo. Everything else is probably less suitable. A top-aligned Label, for instance, is possible, but doesn't usually make much sense.

Now that we have introduced the concept of Containers, and parent and child items, it's time to examine how you **place items in Containers**.

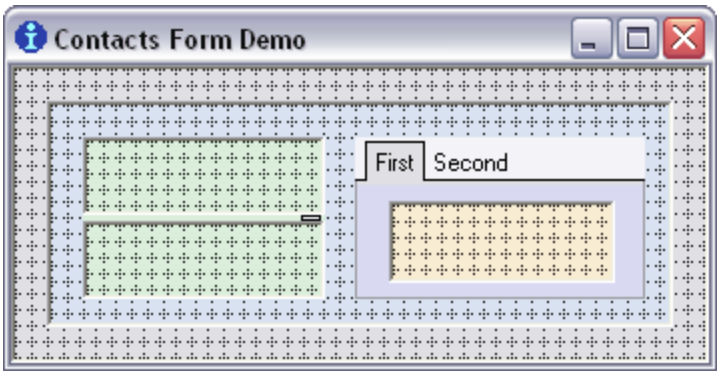
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Actually, there's almost nothing to it. When dragging a field from the Fields grid onto the InfoView, the procedure is exactly the same: you just drag the field and drop it into the Container you want to parent it to (i.e., the one you want to insert the field into). If you have an InfoView and a Panel in it, both of the above are Containers, so you can drag the field onto the InfoView itself or onto the Panel. In the first case, whatever item gets created will be parented to the InfoView and will hence become a sibling of the Panel. In the second case, it will be parented to the Panel and become its child. So, drag-and-drop works the same way.

With other methods of creating items – double-clicking a field or invoking a command such as "Panel" – all you need to do is indicate what your desired destination for new items is. The destination is simply the currently selected Container. If the currently selected item is not a Container (a Label or an Edit Box, for example), Designer will assume that the current destination is the item's parent Container. If nothing is selected at all, this is typically interpreted as if the InfoView itself were selected, so this is what will be the destination.

Some examples follow. Let's assume you have an InfoView with a Panel A, a Split Panel and Page Control within it, and another Panel B in the first Tab Sheet of the Page Control. Plus perhaps some Labels and similar fluff here and there.

To insert a new item, for example a Label, into the InfoView itself, we'd unselect everything by clicking on the InfoView background, and we'd then invoke the Label command.
To insert it into the Panel A, select Panel A or any non-Container item within it – and click Label.
To insert it into the lower Sub-Panel of the Split Panel, select that Sub-Panel.
To insert it into the Tab Sheet – select the Tab Sheet.
To insert it into Panel B – select that Panel or some non-Container item in it.
Well, hopefully you get the idea.



Moving existing items from one Container into another is not very difficult either. Unfortunately, you can't just drag-and-drop them between Containers. What you can do, is use Cut-and-Paste: select the items you want to move; Cut; select the destination Container; Paste. Note that when pasted, the items attempt to maintain their original positions in the source Container. It is sometimes convenient to temporarily resize the destination Container so that the pasted items are at least visible, then move them where you want them, and resize the Container back to its original or desired size. An alternative solution is described in the topic dedicated to Cut-and-Paste. You'll quickly get a hang of it all.

There remains just one last trick we'd like to share with you. It's not a Designer feature per se, but merely a clever way of using the behavior of Containers and items we have described above.

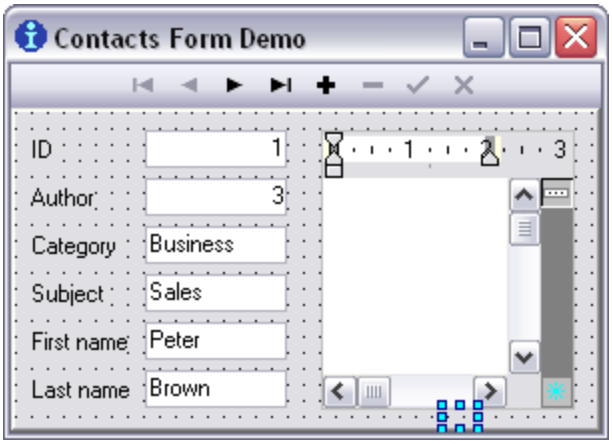
Let's say you have an InfoView form with a relatively simple layout depicted below.

The items on the left keep their sizes and positions constant. The Memo on the other hand gets automatically resized with the form. Why? We have probably set all of its Anchors to On. Fine. But let's see what happens if you shrink the InfoView's width too much. The Memo will simply keep shrinking

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

until there's nothing left. Keep reducing the width of the form, and when we reach the items on the left, the form finally displays a horizontal scrollbar. Everything is behaving correctly, and it's not too bad either, but we can make it better. Wouldn't it be nice if the scrollbar appeared just a little earlier – when there was still some Memo left? That would make the InfoView usable even when shrunk to a microscopic size. But we don't have any direct control over when the scrollbar appears: it does so when the form reaches some non-moving item in it. And that's precisely the trick.

All we need to do is put some non-moving item where we want the scrollbar to show up! There are a few items that would suit this purpose: a Label or a Rectangle, for example. Especially since we can easily make both of them effectively invisible. A Label's text can be set to a single (or several) spaces. Or we can set its font color to the color of the background. Similarly, we can persuade the Rectangle to not display a border. We'll see how all this is done in the next chapter, but the point is: we can place a little invisible item where we want, and its only useful function will be to hit the edge of the form (or whatever Container we are working with) thus persuading it to display a scrollbar.



The above screenshot displays such an item in a position that we consider reasonable. Note that the item is underneath other controls, so all you can see is its selection handles.

We were talking about this "doorstop" item being "non-moving", and by now you probably understand what exactly this means "technically": it's an item with only its default Anchors turned On: Left and Top.

There are some special considerations when deciding on what Container you should place the doorstop item. Generally speaking, it is always true that it is the parent Container of this item that will display its scrollbars when the item no longer fits in it. The rather technical problem (which, incidentally, is not specific to Alventis) is that items anchored right and/or bottom maintain their distance to the visible edge of their Container – irrespective of scrollbars, if any. This may have sounded obscure, and it really is. You don't have to worry about such intricacies though. What's important is the bottom line: always place the doorstop item behind a client-aligned Container. Both the doorstop item and the Container must be siblings, both being parented to their common Container. A simple example would be: the InfoView form (acting here as a parent Container) would have the doorstop item and a Panel. The Panel would be client-aligned thus covering the doorstop item. If that Panel weren't client-aligned but merely anchored on all sides, you'd fall victim to that obscure problem we have mentioned earlier. Try it if you don't believe us.

Creating effective layouts that behave well when you resize the form is half science and half art. It certainly takes a bit of practice and some trial and error. With the unprecedented flexibility of the Layout tools Designer puts at your disposal, you should be soon creating dazzling and fully-resizable InfoViews.

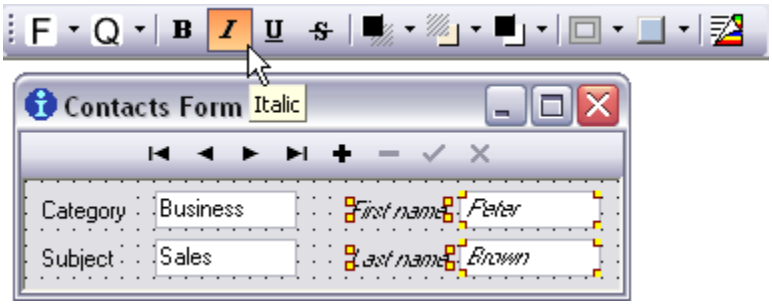
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

3.9 Item Formatting

In the previous chapters we have seen what items we can place on an InfoView form. We have also learned to properly position them so that the InfoView maintains a pleasing and functional Layout when resized. All that remains is to see what other attributes we can adjust for all these items to make them look just the way we want.

In contrast to the vast majority of applications that perform at least some similar tasks, Alventis Designer doesn't force you to set cryptic settings in long lists of properties or learn to use dozens of dialog boxes for each item. The basic principle in Designer is remarkably simple: select the items you want to affect, and apply whatever attribute you wish using an array of mostly familiar formatting tools. If this sounds a bit like formatting text in a Memo, that's precisely how it works!

Want to make a Label italic? Select the Label and depress the Italic button.



Want to set the background color of an Edit Box and a Panel (both together)? Select both and use the familiar Background Color InstaButton. Almost all style adjustments of all items are performed in this fashion and work exactly the same as text formatting works in Alventis Memos. Which is why we won't even bother mentioning 75% of them.

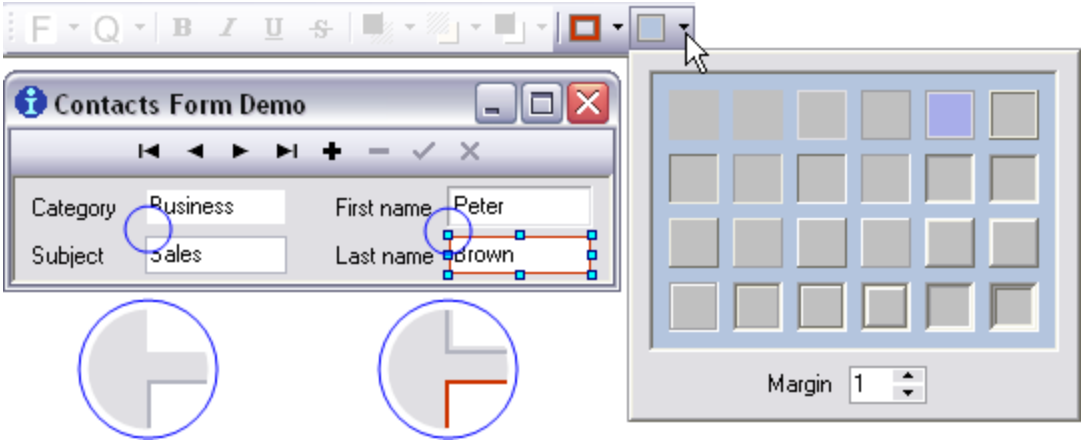
If you need to refresh your memory on how exactly the Font or QuickStyle InstaButtons work, you can always refer to the relevant topics in Alventis Memos section of this Guide. Let's take a closer look at some of the Formatting and other controls we have not seen previously.

In general, Designer tries to keep the number of controls or commands you must use to adjust item attributes to a minimum. The fewer there are, the quicker you'll "learn the ropes". One effect of this approach is that some controls perform slightly different actions depending on what item they are acting upon. The actions will have something in common, so their effect becomes quite natural after a little while, but we wanted to point this out from the onset to avoid surprises.

Border Color InstaButton. It works exactly like all other Colors, except that this one sets the color of the border of items that have one. Edit Boxes and similar items can have various border types (we'll get to them in a moment), and only some of them use the color you select here, so don't be surprised if they seem to ignore your efforts. A Grid doesn't have a border, so this button sets the color of the gridlines instead.

Border Style InstaButton. It has a rather simple dropdown dialog that lets you pick the appearance of the border and adjust its thickness (Margin).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



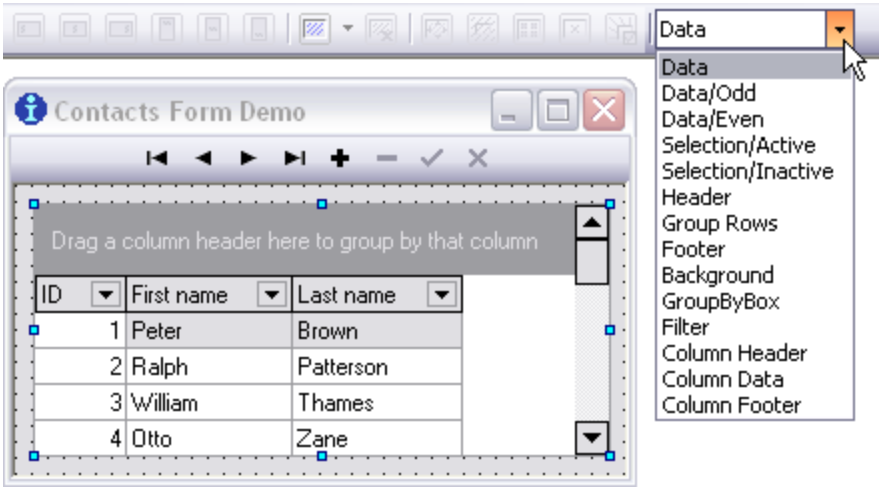
Border Style applies to several items: Panels, Split Panels (and of course their Sub-Panels), Tab Sheets, the InfoView itself (the form's background acts very much like a Panel), Edit Boxes and similar, Rectangle, and even the Memo. The trick is that the above items are quite different, and so are their borders. Panels, for example, support the full array of Border Style settings offered by this dialog. Edit Boxes support only 5 border styles, so a number of similar-looking border styles from the dialog's palette of 24 apply the same Edit Box border style. You'll have to click around a little to see which of the 24 produce what effect.

Panels, Split Panels, Tab Sheets, the InfoView – all support the full array of adjustments.
Edit Boxes and similar support 5 border styles and ignore the Margin.
Rectangle and Memo support only the Margin. They both interpret it as "border thickness + 1": a Margin of 1 means "no border", a Margin of 2 creates a single-pixel border, and so on. It's easy to use even if initially confusing.

Grid Item Selection. Let's get this out of our way right now. You can select the Grid, *that* – you knew. You can also select individual Grid Columns – you've seen how to do this in the InfoView Grids chapter. What we haven't told you is that you can select quite a few other parts of the Grid. Designer refers to them as Grid Items. All available Grid Items are listed in the Grid Item combo box on the toolbar:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

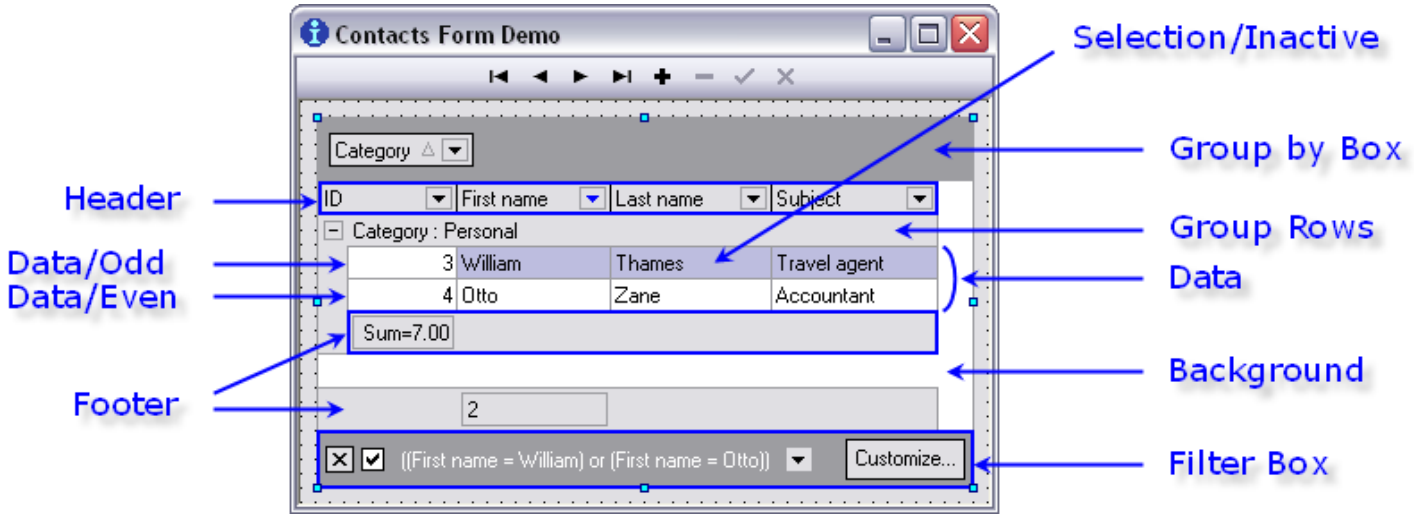
- Data means all Grid Cells
- Data/Odd means cells of the odd-numbered rows
- Data/Even means the even-numbered rows
- Selection/Active means the Grid selection when the Grid has focus
- Selection/Inactive same, but when the Grid does not have focus
- Header is the row of Column Captions
- Group Rows are the Grouping rows, if any
- Footer is the Grid or Group Footer where Summaries are displayed
- Background is the unoccupied empty space of the Grid
- GroupByBox is the Group by Box at the top
- Filter is the Filter Box at the bottom
- Column Header is the Header (Caption area) of an individual Column
- Column Data is the Data area (Cells) of a Column
- Column Footer is the Footer (Summary) area of a Column



The last 3 Grid Items therefore refer to the specific selected Column(s), if any, while everything else refers to the Grid in general.

Selecting what you want is easy. You can simply click the Grid area of interest – and the Grid Item you clicked gets automatically selected in the Grid Item combo box. If it's not the right Item, you can always manually pick the right one from the same combo box. By default, when you click any Cell of the Grid, Designer thinks you are interested in the Data Item. Similarly, clicking the Header row selects the Header Grid Item for you. There's a little trick to let you select the corresponding Column's Item: Alt-click the area you want. Alt-clicking a Column's Header, for example will select the Column itself and set the current Grid Item to Column Header.

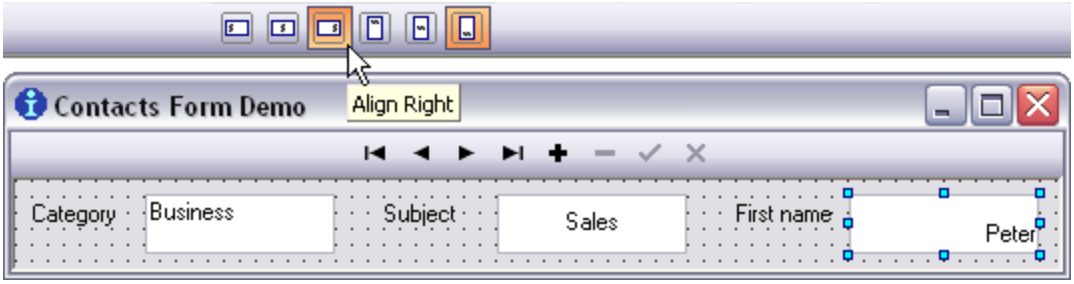
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



It should be noted that while most attributes will apply themselves to the selected Grid Item, some simply don't apply to any, so they may get applied to the Grid as a whole. Border Color is one such example.

Align Text buttons Align Left / Center(H) / Right, Top /Center(V) /Bottom .

These buttons are responsible for setting the alignment of the text within some items that allow text-editing. Such items are Text Boxes and similar, as well as grid columns.



Just select the column(s) you want (make sure the grid itself is selected too), and apply whatever attributes are appropriate. You can even select a few columns and a few Text Boxes – all at the same time. Vertical alignments are typically less useful and only come into play if the text editor (or cell, which in effect is one) is taller than the text.

Enabled. By default, all items are Enabled. Disabling a data-editing item (Text Box, etc.) makes it inaccessible for editing and usually grayed-out. You can usually achieve a somewhat better effect by simply making the item Read-Only (see below).

Read-Only. Applies to various data-editing items and makes it impossible to edit their contents. The item is not grayed out and its contents can still be selected and copied to the clipboard. Both of these factors are usually advantages, so we tend to prefer this attribute to the Enable one (above).

AutoSize. Applies to Labels, Edit Boxes, and similar items that attempt to somehow automatically resize themselves based on their contents. Labels can be resized regardless of this setting but auto-size themselves if their Caption is later edited. Edit Boxes tend to set their own height depending on the

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

chosen font, and don't allow you to set it manually – unless you turn AutoSize Off for them, that is. Once you do (if ever), Vertical text alignments can be adjusted, e.g., to center the text vertically in an oversized Edit Box.

Shadow. A purely visual attribute that applies to various data-editing items. You can simulate a shadow for items that don't explicitly support one (e.g., the Memo) using a properly sized and positioned Rectangle.

HotTrack. Applies to Text Boxes and similar. Makes their border react visually when you pass the mouse over the item.

Page Control Style. A combo box that allows you to set one of 9 Page Control visual styles. Obviously, only applies to Page Controls.

Checkbox Null Style. A Boolean field may obviously contain True or False values, but it can also contain the Null value, which means "no value has been assigned" (this is true of all field Types, by the way). This combo box allows you to select how you want the Null values, if any, to be displayed: as a grayed-out unchecked checkbox, a grayed-out checked checkbox, or as a normal-looking unchecked checkbox (to make it look as if Null were equivalent to False). Note that you can also control whether the user can set the Checkbox's value to Null. You do so using the UniToggle button described below.

RadioGroup Column Count / Currency Precision. Sharing a single SpinEdit control are two completely different attributes. RadioGroup Column Count specifies the number of columns for the Radio Buttons in a Radio Group – and obviously only applies to this item. Currency Precision only applies to Currency Edit Boxes and sets the number of digits you'd like to be editable after the decimal separator. The default is 2, so amounts like 24.95 could be entered (but not 24.951). You should avoid using this "combination" control on both kinds of items it applies to simultaneously, but there's hopefully little risk of that happening.

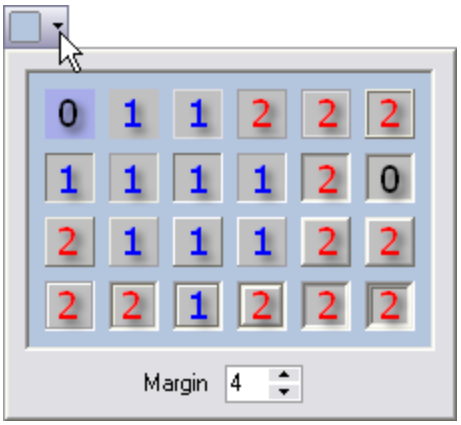
UniToggle. This is a "fun" little control since it applies to quite a few items and toggles something different for each type. We will therefore list the items it affects and what it does:

Label	Toggles WordWrap. When On, Labels will become multi-line if not wide enough to display their entire contents.
Radio Group	WordWrap for Radio Buttons – same as Labels.
Time Edit Box	When On, seconds won't be displayed/edited.
Checkbox	When On, the user can cycle through all 3 possible values: True, False, Null. When Off, just True/False.
Image (static)	When On, stretched images will be stretched proportionally, i.e., maintaining their aspect ratio.
Page Control	When On, Tabs of a Slanted Page Control will be slanted, when Off – only their corners will be "bent".
Multi-Line Edit Box	When On, both scrollbars will be visible. When Off, only the vertical one (and text will auto-wrap at the right margin).
Split Panel	When On, the Splitter has a 3D beveled appearance.
Rectangle	When On, the Rectangle's corners are rounded.
Grid	When On, the columns always occupy the full width of the Grid. When Off, the total width of all columns may be larger or smaller than that of the Grid, which will have a horizontal scrollbar if necessary.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Let's take a break and explore some **Panel attributes**. Firstly, as we have already seen a few times, Split Panels are just "combination" items made up of 3 Panels each. A Tab Sheet is essentially a Panel too, and so is the background of the InfoView form itself. All these Panels share 99% of their attributes, so almost everything we will be talking about applies equally well to all of them. We will point out the rare exceptions to this rule whenever we encounter them.

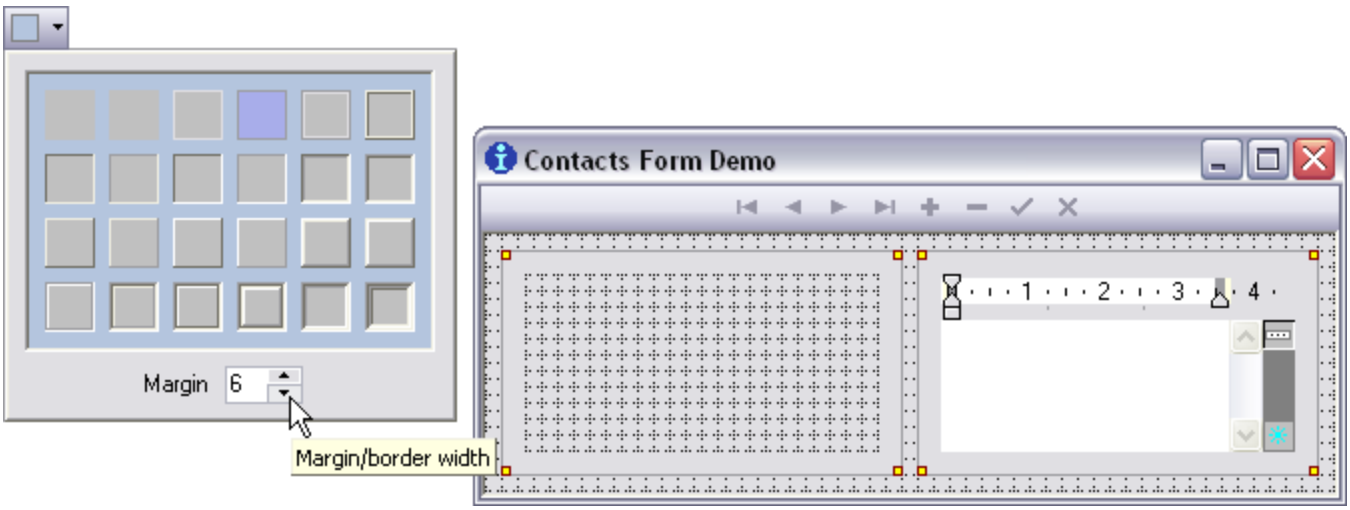
We have already seen that a Panel can have a variety of Border Styles and Margins, but we haven't really explained this.



The Panel Margin is the space just inside its borders that is "off limits" to whatever items may be inside the Panel. Some Border Styles have no associated Margin regardless of what number of pixels you set it to. Some Border Styles do use the Margin. Yet others use a "double" Margin, i.e., make it twice as wide as what you specify.

The screenshot on the right shows you which Border Styles have what Margin (1 means normal and 2 means double).

What good use can you put the Margin to? Actually, not much. Except for one thing. If you put some client-aligned (Alignment set to Client) item inside a Panel with a Margin, you suddenly have a nifty way of controlling how close that item gets to the Panel's edges. Because if the Panel has a Margin, its client-aligned child item will occupy the space within the Margins, but won't venture into their territory.



A Panel can have a Picture Background. In theory, you could achieve a similar effect by placing a static Image item onto a Panel, and making it client-aligned or sizing it whichever way you want. You can in fact do that, and in some cases you may want to. But there's an easy alternative. You can assign the desired picture directly to the Panel itself. The advantage is that there's no longer an Image item, possibly getting in the way of your attempts to select items, etc. There's an even greater advantage

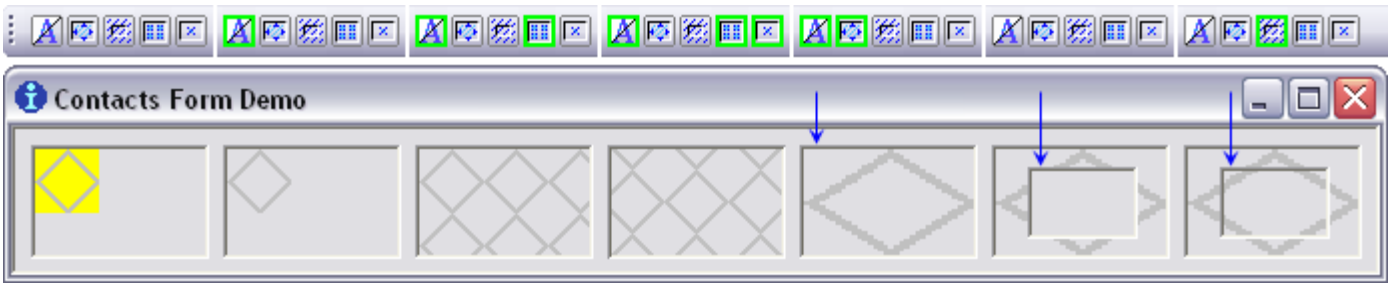
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

that we will discover shortly. But first, how do you assign an image to a Panel? The Panel Background Image button accomplishes this task. Clicking it opens the standard Open Image dialog. You'll notice that as is the case with most such buttons in Alventis, this button has a drop-down menu that holds the most recently used Images. You can clear the assigned Image from a Panel using the Panel Background Clear button. A word of warning: you can use transparent GIF and PNG images as Panel Backgrounds, but in these cases you should also enable the Panel's Transparency. We'll see this attribute in a little while. Otherwise the background is not always redrawn properly, especially if the image is Stretched.

Attribute Inheritance. Many items inherit some of their attributes from their parent Container. Here's how it works. Let's say we have a Panel with a bunch of Labels on it. Labels have such attributes as Font and Background Color. Turns out, so does the Panel. You could have guessed about the Background Color yourself, but Font is less obvious. Imagine what happens if we change the Font of the Panel to Italic. Nothing happens to the Panel itself (since it itself doesn't display any text), but all Labels "sitting" on it suddenly become Italic. Change the Panel's Background Color – and... well, now it seems that it's just the Panel's background that changed, but in reality this is not the case: the Label's backgrounds changed as well, so they simply keep looking "transparent". The above doesn't apply to just Labels of course. The vast majority of available items behave this way, inheriting their attributes from their parent. You can "break the connection" at any time by setting a child item's attribute explicitly. Once you do that, the item becomes independent of its parent's attribute changes and stops "playing along".

You can always make an item use its parent's attributes again using the Clear Style button. It returns most item's properties to their most common default values and the item will once again become an "obedient" child. The Attribute Inheritance often makes it easy to control the appearance of a multitude of items at the same time – without the need to tediously select each and every one of them. Just select their parent Container and adjust its attributes. Setting the Font of the entire InfoView form (when there's no selection), for example, will set the Font of all its child items, and their child items, if any, and so on.
This is why the Font attributes are applicable to Panels.

Now that we have learned that Panels can have Background Images, let's continue looking at item attributes we can adjust.



Transparency. Makes some items transparent. Applies to Labels, Images, and Panel Background Images (now you know why we had to take a break to tell you about them). The way Images (both static ones and Panel Backgrounds) become transparent, if at all, depends on the type of the loaded picture. JPEGs don't support transparency at all. Bitmaps can be transparent or not – depending on the Transparency attribute. When enabled, the Bitmap's lowest/leftmost pixel's color becomes transparent in the entire Bitmap. If the color of that pixel happened to be Yellow, for example, everything that's

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Yellow becomes transparent. When not enabled, the entire image is opaque. GIF and PNG images support their own transparency "explicitly", and the Transparency attribute should be simply "synchronized" with the transparency of the picture file.

Stretch. Makes the picture in Images and Panel Background Images stretched to fill the item's boundaries.

Tile. Applies to Panel Background Images that are not Stretched. Repeats the image to fill the Panel's boundaries.

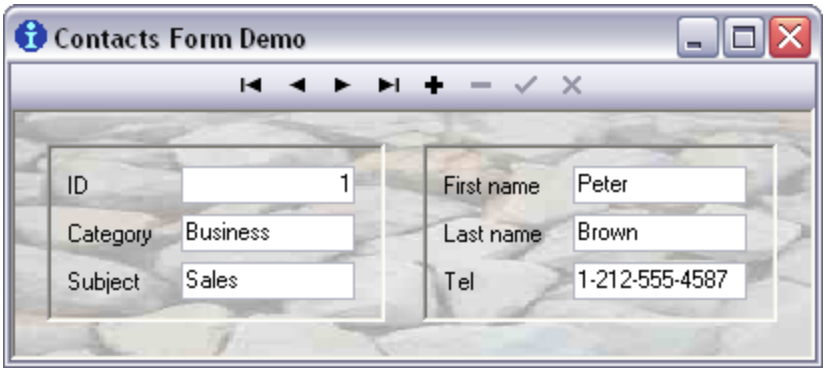
Center. Applies to Panel Background Images and field-based Image items. Make the image centered within them. For Panels, this attribute can be combined with Tiling but not with Stretching, which always takes priority.

See-Through. Applies to Panels. Makes their background transparent as far as their parent Container is concerned. This is a very powerful feature and here's how it works. Let's say we are creating some fairly advanced Layout that requires us to use a few Panels to make things properly react to resizing. Imagine we also want to use an image background for our InfoView.

The problem is quite obvious: the Background Image that we have assigned to the InfoView is at least partially covered by the Panels.

We could try assigning the same Background Image to these Panels, but it will not align properly, certainly not all the time.

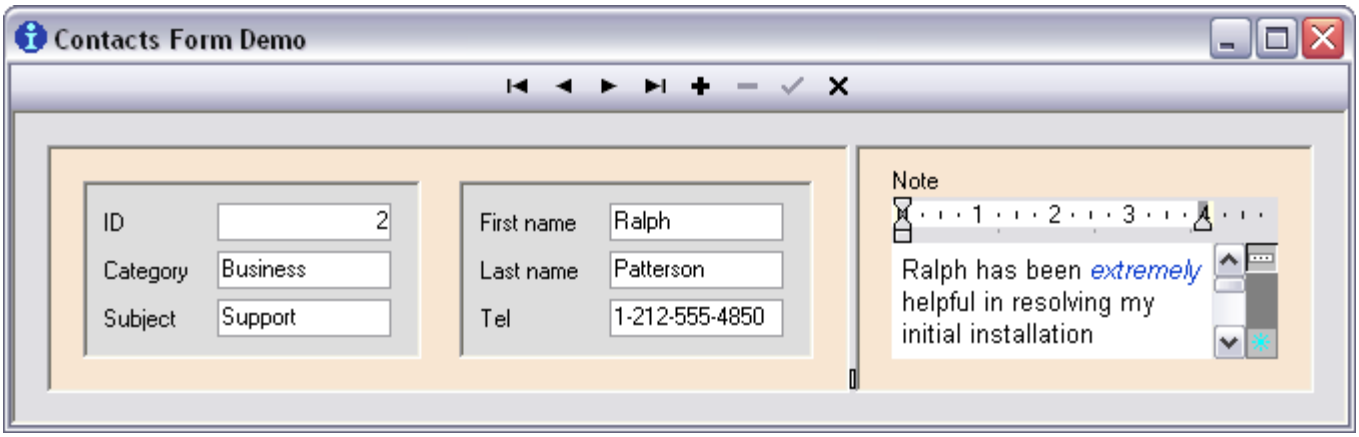
The solution is simple: enable the See-Through attribute for the Panels. This makes them transparent to whatever their parent Container Panel's background is displaying.



The reason we don't simply say that the Panel becomes transparent is that it doesn't: if it completely or partially covers some other item, you won't see that item through the panel – only its parent's background. Since it would be highly unusual for a Panel to cover some item, this is largely not an issue, so you can in fact think of this attribute as making a Panel transparent if you wish.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Zoomable. Applies to Panels, except the InfoView itself. This enables the PanelZoom feature for the Panel. PanelZoom is described elsewhere, but let's see how it works in more detail, specifically, when there are multi-level Containers involved. When PanelZoom is invoked by either double-clicking a particular Panel or using the PanelZoom command, the program starts by getting hold of the double-clicked Panel or the currently-selected item, as the case may be. It then "walks its way" up the chain of parents until it finds a parent Panel with the Zoomable property set to On. This is the Panel that will get zoomed. It may very well happen to be the Panel that was double-clicked, but it could be some parent thereof. For the "end user" of the InfoView in Alventis, there's usually little need to know any of this stuff. He tries to zoom – something gets zoomed (or nothing at all). But it is you, in Designer, who decides which Panels get zoomed, and which simply "pass" the zoom command on to their parent Panels. This is a confusing subject, so let's clarify it with an adequately confusing example.



In the form above, there's a Split Panel with two columns of items in one Sub-Panel, and some other stuff we don't care about in the other. The two columns of items have been made to resize proportionally by placing each column on its own Panel and making both Panels Proportional. Why all this was done is not really important though. What's important is which Panel(s) we should make Zoomable. We could make them all Zoomable of course. But then, if the user has the cursor in an Edit Box in the left column, and invokes the PanelZoom command, only that single column's Panel would get zoomed, since this would be the closest Zoomable parent Container of the focused item. If we really wanted to make things nice, we'd want both columns to zoom together. To achieve this, we would make the two Proportional Panels non-Zoomable, and we'd make their common parent Container – the first Sub-Panel – Zoomable. That's the one that will get zoomed then, together with both of its child Panels and their two columns of Edit Boxes. It's up to you to decide which Panels, if any, you want to zoom under what circumstances, and now you know how the applications decide which one to zoom.

3.10 Navigation and Tab Order

An effective InfoView must be as keyboard-friendly as it can be. To Designer, this means two things:
- Properly setting the Tab Order
- Properly setting Label's Accelerators

The Tab Order and Label's Accelerators are two independent mechanisms that make navigation of the form easier using the keyboard – even in this day and age of the reign of the Mouse.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

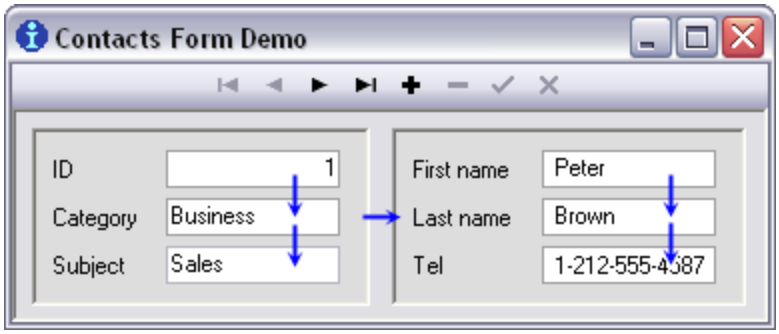
The **Tab Order** is so obvious, yet it is often overlooked even in some very serious applications. Tab Order is the order in which the focusable items on an InfoView form are "visited" when one clicks the Tab key. To adjust the Tab Order, Designer offers you not one, but two methods.

Tab Order Direct. This is exceptionally simple but requires a little concentration on your part. With no items initially selected on the InfoView, select all focusable items (Edit Boxes and the like) in the desired Tab Order – one-by-one: select the first one, and then just keep Shift-clicking the rest of them. Once all necessary items are selected, click the Tab Order Direct button. That's it. The Tab Order is set to the order in which you selected the items.
You can use the above procedure on a subset of items. For example, if you have items that currently have the incorrect tab order A, C, D, B, E, F, ..., you could re-arrange the order of just the C, D, B group by selecting these 3 items in the right order B, C, D, and clicking the Tab Order Direct button.

The above "direct" procedure may work well for relatively simple InfoViews, but sometimes you may wish you had a way to just see what the current Tab Order is and make small adjustments to it. Your wish is our command, and the second method of setting Tab Order offers you precisely that and then some. But before we delve into it, let's discuss a little Tab Order problem.

Tab Order and Containers. The problem is this: Tab Order works within Containers. That sounded menacing but not very clear, so let us explain.

Imagine you have Panel A with a few Edit Boxes, and next to it – Panel B with a few more Edit Boxes or some such focusable items.



The way Tab navigation works, you have a few choices as far as the Tab Order:
- you can choose which Panel gets visited first
- you can choose the order in which the Edit Boxes in Panel A are tabbed through
- you can choose the same for Panel B's Edit Boxes

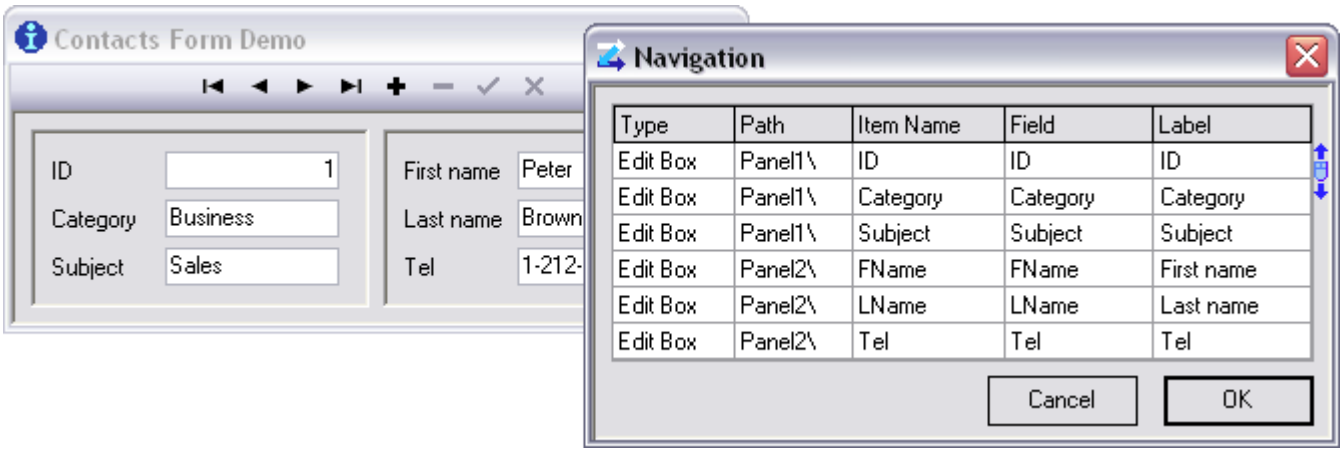
What you cannot do, is persuade the form to make the Tab key move the focus directly from one Panel's Edit Box to an Edit Box of the other Panel – in a zigzag fashion. You have to go through all items of one Panel, and only then are you allowed to move on to the next Panel. As a matter of fact, you may never notice this because Panels themselves do not receive focus, but Panels (and other Containers) have a particular Tab Order associated with them – just as Text Boxes and some other focusable items do.

We had to mention the above problem or issue simply because you are very likely to quickly run into this limitation, and get upset, annoyed, confused, and perhaps even become quite rude with your computer. There's little reason for any of the above though. For one thing, if some items end up grouped together in the fashion we have illustrated above, it's not all that dreadful if Tab navigation between them works in "columns" order instead of the admittedly more natural zigzag one. Secondly, there's a solution even for this insurmountable problem. If you absolutely insist on the zigzag Tab Order, just place the 6 Edit Boxes on 6 Panels! It may seem like too much trouble and in most cases it

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

probably is.

The Navigation button invokes the dialog box depicted below:



The list shows all focusable items of the InfoView in a grid with the following columns:

- Type is the type of the item
- Path shows what Container(s), if any, the item is in
- Item Name is the name of the item/control
- Field shows what data field the item is "attached" to
- Label shows the Label item that is set up to focus this item – a feature we'll discuss shortly

Adjusting the Tab Order is as simple as drag-reordering the corresponding records of the grid: just drag the ones you want where you want them. You can drag more than one item at a time.
And this is when you need to remember the Tab Order / Container limitation we have described earlier. The list displays all items and, alas, allows you to re-order them whichever way you want, including a zigzag fashion that will not work. Applying such an "illegal" Tab Order will essentially make Designer do its best to make it "legal", i.e., it will attempt to honor it the best it can while complying with the Tab Order within Container rule. This will produce the effect of the Containers being visited in the order in which their first child item has been listed by you. If you would like to make sure you are looking at a proper "legal" Tab Order, it is your responsibility to list child items consecutively "within" their parent Container. As we said, this is not important to be able to set the best reasonable Tab Order, but you may want to do it this way simply to avoid surprises.

Label's Accelerators. A Label has secret powers. Well, at least one: it can focus an item. For this to work, two things must be set up:

- The Label must have a valid Accelerator character. An Accelerator is the first character in the Label's Caption that follows the ampersand "&". Windows underlines such characters (at least when you press and hold the Alt key, depending on your Windows settings). When you click Alt- Accelerator (whatever character that happens to be), some action takes place. If the Accelerator is that of a Checkbox, it's as if you clicked it. Same for Radio Buttons. It's similar for Labels too, except that they don't "get clicked" – they perform a function that depends on the other, second thing to be properly set:
- The Label must have an associated focusable item. A label that has such an item will focus it when its Accelerator is invoked. The Label column of the Navigation list lets you associate items with Labels. To do so, you simply pick the Label you'd like to be the item's "focuser" from the item's combo box. Since

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

a Label can only focus a single item, it can't appear twice in the list, so an attempt to assign the same Label the second time will simply "steal" it from its original item.

The Label column does not indicate if a given Label has a valid Accelerator character. You can verify and adjust the Accelerators directly in the InfoView by editing item Captions as necessary. An item, such as a Checkbox, that has its own Caption can usually focus itself with no need for a Label's help. Labels that are auto-created when you create items based on fields are pre-set to focus their corresponding item. You can edit this for any label if necessary: there's nothing else that makes these default Labels "special". If you ever want to have a Label with a literal ampersand character, you must double it up so that it is not interpreted as an Accelerator "marker". For example, you could create a Caption "Cats && D&ogs" which would be displayed as "Cats & Dogs".

Accelerator Conflicts. You must watch out for two possible sources of conflicts between Accelerator characters. One is within the same InfoView. No two items should have the same Accelerator. If they do, it is most likely that only one of them will work. The second source of conflicts is the main menu of the application. If both the menu and an InfoView end up with the same Accelerator character, the menu "wins", and the InfoView's one is ignored. Since InfoViews and Accelerators are conceived to work in Alventis, it is Alventis' main menu you should be worried about – not Designer's. Since nobody will prevent you from setting a wrong Accelerator, you have to approach this systematically. Start by making a list of Accelerators taken by Alventis' menu (E F G H I O T U V W). You can now assign unused letters to your Captions. You can also use numbers or whatever else is accessible with the Alt key: if you "run out" of all the common stuff, you can even venture into such exotic Accelerators as the dollar sign "$" and similar. One would have to invoke these with both the Alt and Shift keys though: Alt-$ is equivalent to Alt-Shift-4, for example. Note that you can't use this trick to distinguish between uppercase and lowercase letters. In closing, here's a little tip on how you might proceed with your Accelerator choices: start with the shortest Captions with the most common letters on the one hand, and with the most infrequently used letters (in all Captions) on the other – and attack the problem from both of these directions. You'll see why this makes sense when you start assigning Accelerators to your 27 items...

3.11 Calculated Fields

Warning: Advanced Material

Alventis and Designer support Calculated Fields via expressions or formulas written using the SQL language. To create a Calculated field, all you need to do is create a String field as you normally would, and specify the formula for it in the Formula cell of the Fields grid. In general, a field can be either a "real" physical field that ends up saved to the physical data table – or a Calculated field that is not saved to the table but is merely maintained "virtually", in-memory only, using the formula you specify. It is the existence of the formula that determines whether the field is physical or Calculated. All formulas are deemed to produce a String result, so only String fields can be Calculated and have a formula.

Formula Syntax. Alventis uses the formula by inserting it into a SELECT query statement: SELECT <formula> AS CalculatedFieldName FROM TheFieldsTable. For example, if the formula is "FName + ' ' + LName", the query becomes "SELECT FName + ' ' + LName AS CalculatedFieldName FROM TheFieldsTable". You can therefore use whatever SQL syntax is legal for specifying the calculated "column" value, including the full array of SQL operators and functions available to you. Please refer to the SQL Reference if you need more information.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Note that it is possible to produce multi-line "string" output by concatenating lines with the CR/LF characters like this:
'Text of first line' + #13 + #10 + 'Second line'
Such calculated String fields are great candidates for the Multi-Line Edit Box item (which is its primary intended purpose).

It is possible to "convert" an existing field between a physical and Calculated state. That is, a field that already exists in the table can become Calculated. Just supply the formula and save the Table. The physical field will be deleted and all the data it may have been storing will be lost, so exercise caution. Removing a formula from a Calculated field converts it to an ordinary physical one.

3.12 Validation

Warning: Advanced Material

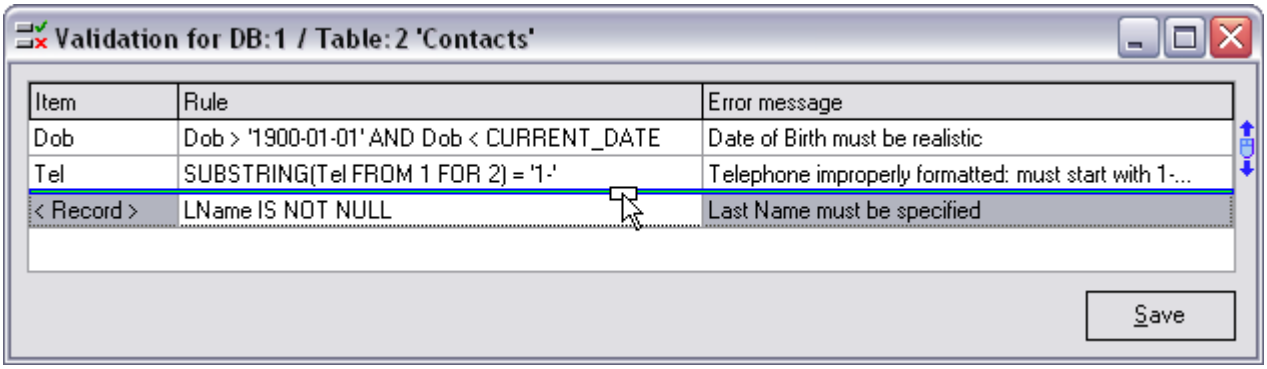
Alventis and Designer support Field-Level and Record-Level Validation. Basically, you can specify a condition that must evaluate to True for an individual field's value to be accepted during record editing – or the whole record to be considered valid when posting it. The validation conditions are specified using the SQL language. Technically speaking, the condition ends up being used as the expression in a query's WHERE clause, so you can use whatever SQL syntax is legal for this clause.

To edit Validation conditions, you must indicate which table's Validation you want to edit using any of the following means:

- Highlight the desired table in the Tables grid
- Highlight any field of that table in the Fields grid
- Focus an InfoView of that table

Once this is done, click the Validation button to open the Validation dialog. Note that this dialog is modeless to allow you to switch to another form in Designer while leaving it open.

The Validation dialog is remarkably simple. All it has is a list of conditions in a grid.



Create new conditions by inserting new records in the list. Delete them by deleting the corresponding record. Re-order the conditions using drag-reordering.
Item specifies what the condition applies to. This can be an individual field or the entire record.
Rule must be set to the desired validation expression.
Error message is the message you want displayed to the user if the validation of this condition fails.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The field-level expressions are evaluated when the user edits a field's value and tabs-out of the field (this action posts the value to the field). If the validation fails, the user receives the error message you specified for the specific failed condition, and he is prevented from leaving the field until he fixes the problem.
Record-level expressions are evaluated when the user attempts to post the whole record. The principle of operation is same as described above.

If multiple validation rules exist for a single posting operation (be it field-level or record-level), they are all applied in the order in which they are listed in the Validation dialog. The first rule that fails displays its error message and aborts the operation. You may therefore find it useful or convenient (both for you and for the end-user) to specify multiple rules per item instead of a single "combined" rule.

For example, you could have the following two record-level rules for the Contacts table:
LName (last name) must not be empty
DateOfBirth must be between 1950 and 2000

You could then express each rule individually with the following setup:

Item	Rule	Error message
LName	LName IS NOT Null	Last Name must be specified
DateOfBirth	DateOfBirth >= 1950 AND DateOfBirth <= 2000	DOB must be in the 1950-2000 range

This "condition separation" will make it quite clear to the user what exactly his problem is. You could of course specify both conditions in a single Rule, but the user would have to guess which problem he hit.

Note that the above example is merely for illustration purposes: in reality, since each of the above rules applies to a single field, it would probably make more sense to make them field-level conditions. And this is pretty much how you should approach validation:
- Validation rules that validate a single field's value should probably be field-level rules
- Validation rules that use more than one field are good candidates for record-level rules

3.13 Relational Databases Design

Alventis and Designer support creation of relationally-linked tables and InfoViews based on them. The overall concepts behind Relationality are explained in the chapter on Databases. Here, we will examine how this powerful feature works in Alventis and what you need to do in Designer to implement it.

You probably remember that to establish a relational link, a record of table A must be referencing another record from table B. This is achieved by having a special Pointer type field in table A. This field "points" to records of table B by storing their ID values. To establish a link between tables A and B then, all we need is a single Pointer field in A pointing to table B. Note that we are not using particularly precise terminology when we say the above. It would be more correct to say something to the effect of "value of the Pointer field from each record of table A points to some record from table B", but this is quite a mouthful, so we won't be very strict with our wording.
The field Type called Pointer is precisely the type we need.

Using an example very similar to the one from the Database Principles chapter, we must then ensure that the Contacts table has a Pointer field pointing to the States table. This field can be called anything

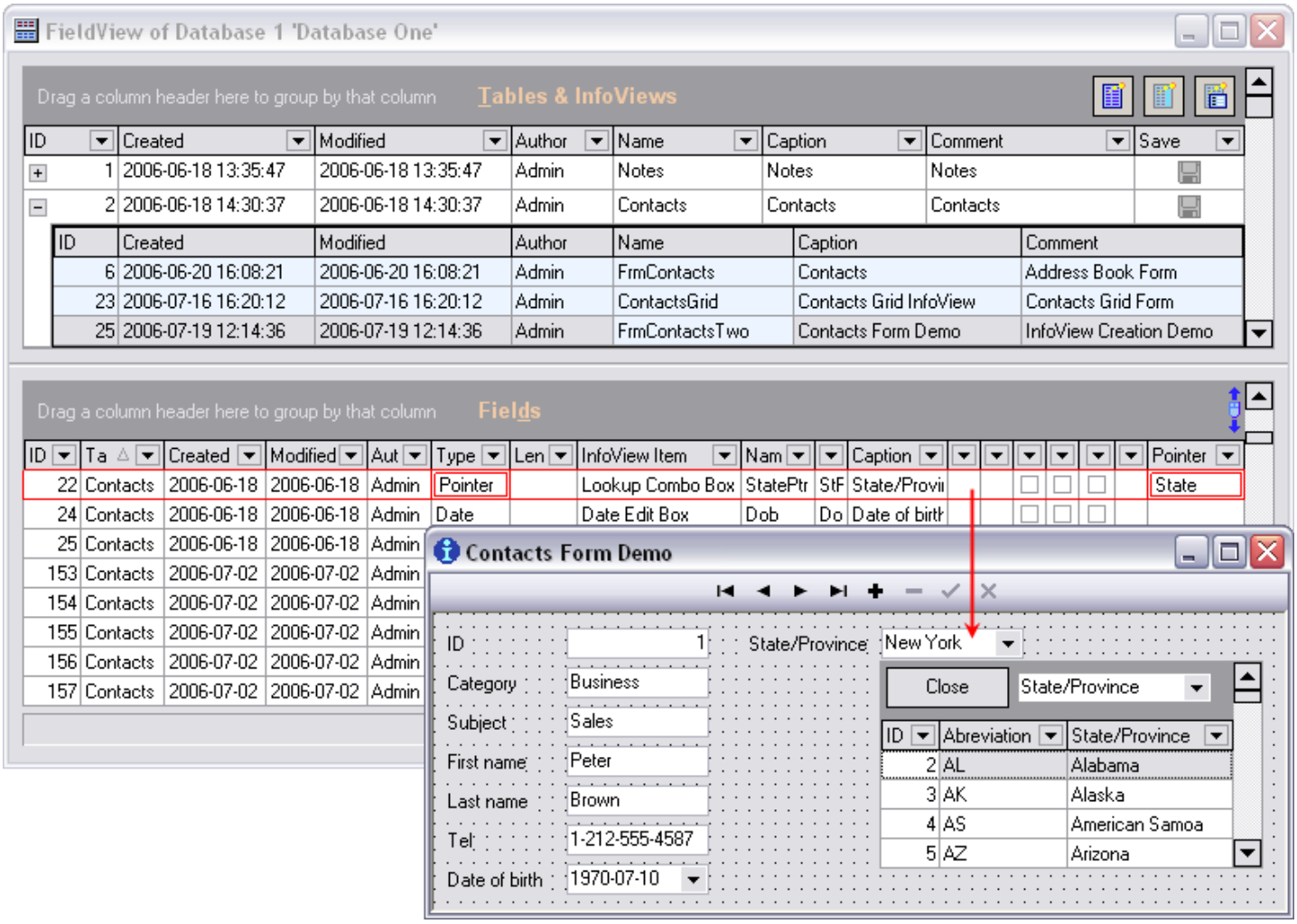
Copyright © 2002-2007 Alventis Corporation. All rights reserved.

we like, so we'll call it StatePtr. In the Contacts table, we shall create a new field, set its Type to Pointer, set its Name to StatePtr, and set its Pointer Target to States. That's it. Note that there's nothing that needs to be changed in the States table, which remains quite oblivious to the fact that some other table is now pointing to it.

With the relationship established at the table/field level, we must do the same at the InfoView level. There are two distinct kinds of Relational InfoViews that we can create, and we'll examine each one below.

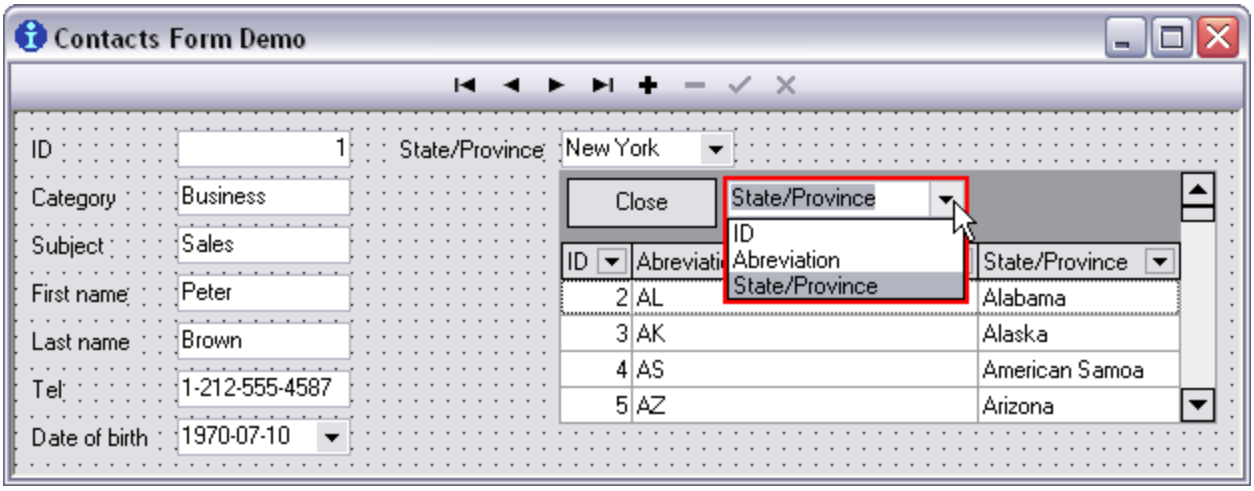
Lookup Relational InfoView. This is the most obvious kind. Basically, we are basing the InfoView almost entirely on the main Contacts table: all fields that we place on the InfoView come from it. So, we end up with a rather ordinary looking InfoView with a bunch of Text Boxes for FName, LName, DateOfBirth, and the like – plus one new item for the StatePtr Pointer field: the Lookup Combo Box. As described in the Database Principles chapter, the Lookup Combo Box is a bit of a hybrid between an ordinary Combo Box and a grid. As a Combo Box, it is used to display and select the State that each Contact record points to (i.e., references or belongs to – whichever terminology you prefer), which is its main relational operation. As a grid, it allows you to view and even edit the records from the States table directly in the drop-down grid. You can also easily open the desired States record by Alt-double-clicking it or hitting Alt-Enter when it is focused. Since the Lookup Combo Box has pretty much an entire grid attached to it, we must populate that grid. Since this grid is supposed to be displaying fields from the States table, this is the table whose fields we must drag into this grid (or "send" them there by double-clicking them).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.


FieldView of Database 1 'Database One'
Drag a column header here to group by that column
Tables & InfoViews
ID Created Modified Author Name Caption Comment Save
1 2006-06-18 13:35:47 2006-06-18 13:35:47 Admin Notes Notes Notes
2 2006-06-18 14:30:37 2006-06-18 14:30:37 Admin Contacts Contacts Contacts
ID Created Modified Author Name Caption Comment
6 2006-06-20 16:08:21 2006-06-20 16:08:21 Admin FrmContacts Contacts Address Book Form
23 2006-07-16 16:20:12 2006-07-16 16:20:12 Admin ContactsGrid Contacts Grid InfoView Contacts Grid Form
25 2006-07-19 12:14:36 2006-07-19 12:14:36 Admin FrmContactsTwo Contacts Form Demo InfoView Creation Demo
Drag a column header here to group by that column
Fields
ID Ta Created Modified Aut Type Len InfoView Item Nam Caption Pointer
22 Contacts 2006-06-18 2006-06-18 Admin Pointer Lookup Combo Box StatePtr StF State/Provii State
24 Contacts 2006-06-18 2006-06-18 Admin Date Date Edit Box Dob Do Date of birth
25 Contacts 2006-06-18 2006-06-18 Admin
153 Contacts 2006-07-02 2006-07-02 Admin
154 Contacts 2006-07-02 2006-07-02 Admin
155 Contacts 2006-07-02 2006-07-02 Admin
156 Contacts 2006-07-02 2006-07-02 Admin
157 Contacts 2006-07-02 2006-07-02 Admin
Contacts Form Demo
ID 1
Category Business
Subject Sales
First name Peter
Last name Brown
Tel 1-212-555-4587
Date of birth 1970-07-10
State/Province New York
Close
State/Province
ID Abreviation State/Province
2 AL Alabama
3 AK Alaska
4 AS American Samoa
5 AZ Arizona

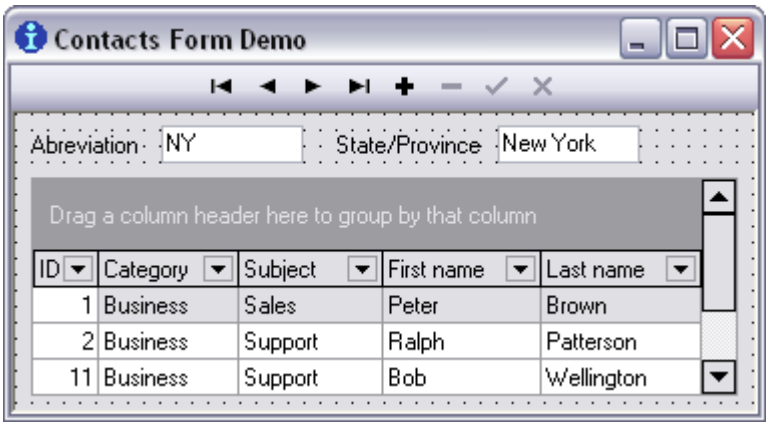
The Lookup Combo Box's grid opens automatically when the item is created. If it's not open, you can do so by double-clicking the Lookup Combo Box. Once you're done with the grid, you must close it using the Close button on top of it. There's just one other element of interest here. Imagine you have added 3 fields from the States table to the grid (ID, Abbreviation, and FullStateName – or something similar). When the grid is not dropped-down, the Lookup Combo Box only displays a single States field value – but which one? If there's more than one column in the grid, we must explicitly tell it which one we want to see in the Lookup Combo Box in its closed state. We do so using the little combo box next to the Close button.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Reverse Lookup Relational InfoView. This is only slightly trickier. The fundamental idea is to base the InfoView mainly on the States table, and for each State record display all Contacts records that point to it.

The above idea gives us a strong clue as to what we can expect to find in such an InfoView. It's mainly based on the States table, so it shouldn't be too different from any non-relational form based on that table. This suggests that our form will probably have a bunch of fields from the States table scattered all over it. And how would such an "ordinary" form display a list of anything, and in particular of Contacts records? Well, a list seems to suggest a grid. Turns out this is exactly what such an InfoView looks like.


Contacts Form Demo
Abreviation
NY
State/Province
New York
Drag a column header here to group by that column
ID Category Subject First name Last name
1 Business Sales Peter Brown
2 Business Support Ralph Patterson
11 Business Support Bob Wellington

If we were to spell-out a "definition" of the Reverse Lookup InfoView, it would go something like this: it is an InfoView that contains one or more items implementing fields from the pointed-to table, and a grid implementing the pointing table.

In the above example, the "pointed-to" table is States, and the pointing one is Contacts (Contacts points to States, remember?). Note that the above "definition" does not specify what items are implementing the pointed-to table's fields. This is because it absolutely doesn't matter. It can be an array of Text Boxes or a grid (this one based on the pointed-to table and not to be confused with the other one that lists records from the pointing table), or even a combination of both. What is mandatory is that the pointing table be represented by a grid. It is also worth noting that nothing in the above "definition" suggests the order in which fields should be added to the InfoView and the grid. This is not

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

a careless omission: the order does not matter. Designer allows you to start the InfoView as a completely "innocent" non-relational form, based on the pointed-to table, and then suddenly turn it into a relational one simply by adding a grid based on the pointing table. Conversely, an InfoView can start its life with just a single grid based on the pointing table (only a grid, no other field-based items!), and then become relational the moment you add a field from the pointed-to table. Either way, we end up with the very same result.

3.14 New Object Wizards

There are three Wizards in Alventis Designer that can assist you in creating the three main types of objects:

New Table Wizard
New Field Wizard
New InfoView Wizard

All of these Wizards are quite similar, and can be invoked either from the File menu or directly from a FieldView form's top-right corner. The Wizards are fairly self-explanatory, so you should have no trouble with them.

The Wizards perform the exact same tasks that you can accomplish by simply inserting a new record into the corresponding grid in the FieldView form. The FieldViews topic describes this in more detail, as well as all the items and options you may encounter in the Wizards.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Additional Information

4 Additional Information

4.1 Server Installation and Setup

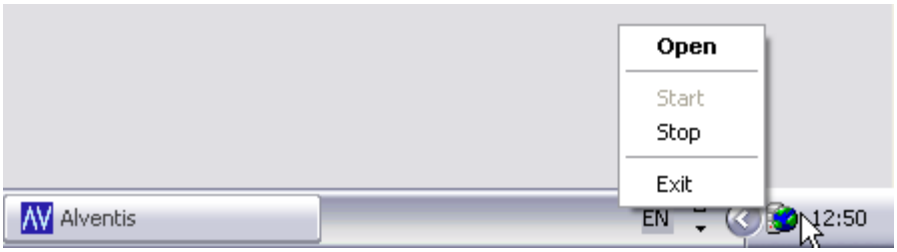
The contents of this chapter is only relevant to the Enterprise Edition of Alventis.

Alventis Server is a standalone application which implements the Server side of the Client-Server mode of operation of Alventis.

The Server application is part of the DBISAM database engine manufactured by Elevate Software (www.elevatesoft.com). It is remarkably compact and easy to set up. As easy as launching a single executable. This chapter will be largely quoting Elevate Software's instructions.

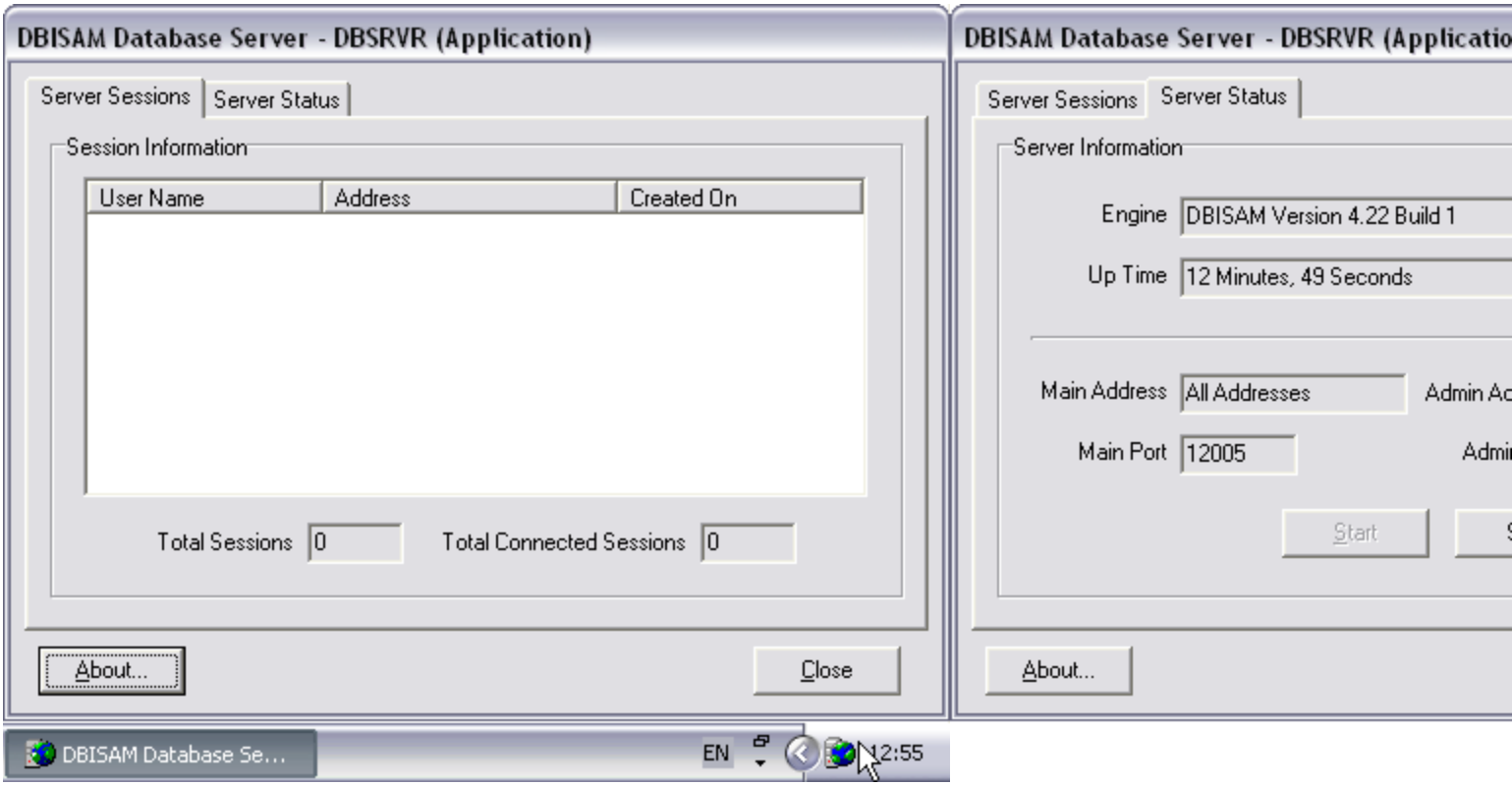
The Server consists of one small executable file dbsrvr.exe accompanied by a same-name dbsrvr.ini configuration file. There is no installation or deployment procedure to follow. Place the above two files in an empty (preferably) directory of your choice, and you're ready to launch the Server.

When running in normal interactive mode, the Server appears as an icon in the System Tray. Right-clicking it displays a context menu with a few obvious entries.



Double-clicking it displays a status window.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Server Configuration. The dbsrvr.ini configuration file specifies the Server's setup and configuration parameters. It's a regular Windows INI file, and as such can be edited in Notepad or any other plain-text editor. When launching, the Server tries to load its settings from an INI file with the same name as the executable and the .ini extension. If such a file does not exist, the Server starts with default settings which would be fine for most circumstances. You can also launch the Server with command-line switches that would specify the desired parameters. Each switch has an equivalent INI file setting, and they are listed in the following table:

/sn Server Name. Specifies the user-defined Server name that will be used to identify the Server to remote sessions. You must enclose the Server name in double quotes if there are spaces in it. The Server name is informational only.

/sd Server Description. Specifies the user-defined Server description that will be displayed in the caption of the Server's user interface. You must enclose it in double quotes if there are spaces in it. The Server description is informational only.

/sa Server Address. Specifies the main Server IP address that the Server will bind to for accepting inbound data connections. The IP address must be specified directly after the /sp switch using dot notation (i.e. 192.168.0.1). The default IP address that the Server will bind to if this switch is not specified is all IP addresses available on the machine. Using this option will cause the Server to only listen on the specified address. This means that it will no longer listen on the local loopback 127.0.0.1 address.

/sp Server Port. Specifies the main Server port that the Server will bind to for accepting inbound data connections. The port number must be specified directly after the /sp switch. The default main port that the Server will bind to if this switch is not specified is 12005.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

/aa Administration Address. Specifies the administration Server IP address that the Server will bind to for accepting administrative connections. Same considerations apply as for Server address.

/ap Administration Port. Specifies the administration Server port that the Server will bind to for accepting administrative connections. The port number must be specified directly after the /ap switch. The default administration port that the Server will bind to if this switch is not specified is 12006.

/st Server Thread Cache Size. Specifies the main Server thread cache size. The thread cache size controls how many threads the Server will cache in order to speed up connect/disconnect times. The thread cache size must be specified directly after the /st switch. The default main thread cache size that the Server will use if this switch is not specified is 10.

/at Administration Thread Cache Size. Specifies the administration Server thread cache size. Similar to "/st" above, but this controls administration threads and defaults to 1.

/cf Configuration File. Specifies the Server configuration file name. The configuration file is where the Server stores its internal configuration information including databases, users, permissions, etc. You must enclose the configuration file name in double quotes if there are spaces in the configuration file name. Do not specify a file extension for the file since the Server always uses the ".scf" extension for all configuration files. The default configuration file name that the Server will use if this switch is not specified is "dbsrvr".

/cp Configuration Password. Specifies the Server configuration file password. The configuration file password is used to encrypt the contents of the configuration file. You must enclose the configuration file password in double quotes if there are spaces in the configuration file password. The default configuration file password that the Server will use if this switch is not specified is "elevatesoft". Note: Do not lose this password. If you do, the Server will not be able to read the configuration information.

/en Encrypted Only. Specifies that the main Server will require encrypted connections only. Note that the administration Server always requires encrypted connections.

/ep Encryption Password. Specifies the password to use for encrypting all data between any remote sessions and the Server. This switch can be specified without the above /en switch to change the password for encrypted connections to the administration Server only. If combined with the above switch, this switch will change the password for encrypted connections to both the main Server and the administration Server. You must enclose the encryption password in double quotes if there are spaces in the encryption password. The default encryption password that the Server will use if this switch is not specified is "elevatesoft". This password must be correctly specified in the Remote Encryption Password column in the Servers grid of the Database Explorer form.

/ni No User Interface. Specifies that the Server should not display any user interface at all. When this switch is used you must use the task manager to stop the Server process. This switch only applies to running the Server as an application and does not apply to running the Server as a service, which instead uses the /nointeract installation switch to determine if it should interact with the Windows desktop in any fashion. The Server defaults to having an interface when it is run as an application.

/al Append to Log. Specifies that the Server should append to any existing Server log file when the

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Server process is started. The default behavior is to overwrite the log every time the Server process is started.

/lf Large File Support. Alventis does not support this feature, so you should never use it. By default, the Server supports a maximum file size of 4Gb for any physical file (data, indexes, or BLOBs) that is part of a logical table and a maximum of 200 million records within a given logical table. Enabling this parameter sets these limits to 128Gb and 1 billion records respectively. Once again, we only mention this as a matter of "full disclosure", but this feature is not supported by Alventis.

The Server can be run as a normal application as we have discussed above. The Server can also be run as a Windows NT/2000/XP service. To do so, you must install it as a service by running the Server with the /install command-line switch: dbsrvr.exe /install. If you wish to install the Server so that it does not interact with the desktop at all, which is required in instances where the current user will be logged out of the system, you should use the /nointeract flag along with the /install command-line switch: dbsrvr.exe /install /nointeract. This will install the service as a non-interactive service and the database server will not display a user interface when it is started.
To uninstall the Server service run it with the /uninstall switch: dbsrvr.exe /uninstall

The only difference between starting the Server as a normal application and as a service is in the way the switches are specified. When starting it as a normal application, you may specify the switches directly on the command-line that you are using to start the Server. For example, the following command will start the Server using port 13000 for the main port and 13001 for the administration port: dbsrvr.exe /sp13000 /ap13001. When starting the Server as a service, you may specify the switches via the Startup Parameters in the Services dialog of the Control Panel, or directly on the command-line that you are using to start the database server with the net start command. For example, the following command will start the database server as a service using port 13000 for the main port and 13001 for the administration port: net start dbsrvr /sp13000 /ap13001. Note that in order to start the Server as a Windows NT/2000/XP service the Server must have already been installed as a service using the /install command-line switch. To start the Server as a Windows NT/2000/XP service automatically when the operating system boots up, you must use the INI file since there is no other way to provide parameters in such a scenario.

You can run multiple instances of the Server on the same physical machine. When running it as an application, it is sufficient to make a second copy of the Server's executable (and INI file) in a second directory, thus effectively creating two independent Server installations. When running the Server as a service, this is insufficient since the service name is set to match the name of the Server's executable, and you can't have two services with the same name. You must therefore rename one of the Server executables (together with its accompanying INI file and other files, such as settings, if already created).

Aside from the Server configuration settings listed above, all other Server settings (Databases, Users, Security, and a few options) are controlled via the Alventis Administration interface.

4.2 Database Principles Overview

In this chapter we will present a very brief and "unscientific" overview of what Databases, Tables, and Records are. This is a vast subject that entire books are dedicated to, and we can only hope to "scratch the surface" – enough for you to understand some basic principles behind Alventis' information

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

handling.

To start with at least some sense of perspective, let's take a look at a Table first. Here's an example of some information about people:

Jack Thompson 612-555-8478
John Barclay 614-555-8114
Helen Bolton 231-555-1421

The above list could be something jotted-down on a piece of paper, but there's clearly a pattern here that makes it more than just "a list". The pattern is that we keep seeing the same "bits" of information about each person listed: first name, last name, and what looks like a telephone number. This stands in stark contrast to another possible list on the other side of the same piece of paper:

Tom Greene 332-555-6478
Bladerunner 1982 Ridley Scott 10 Philosophical Sci-Fi
4 Mars

What's listed above is also "a list" and it certainly contains some information. The problem with this one is that we can't make out a pattern: every line is different, seems to "talk" about different things, and so is quite disjoint from its neighbors.
The first list is a good candidate to be represented by a Table. The second list, alas, is not.

Back to the first list then, let's define what we see in a bit more detail. We see 3 lines with repeating patterns of data: first name, last name, telephone number. We can even write this on top of the list and format it nicely like this:

First name	Last name	Telephone number
Jack	Thompson	612-555-8478
John	Barclay	614-555-8114
Helen	Bolton	231-555-1421

Guess what, we have just created a Table! Well, to be a "real" Table in the pure Database sense, we'd need to put all of the above into some data file, but let's not worry about such details for the time being. The above organization of information essentially constitutes a Table, that's all that matters. It's not surprising, by the way, that the above also just happens to be a very different kind of table, one inserted in some surrounding text by a word processor, what we refer to as Memo Table in the rest of this manual. This is because the organization of data in these two kinds of tables tends to follow the same principles:

- Data is organized in columns and rows
- Each column is homogeneous, i.e., it displays the same "bit" of information, be it first name or telephone number
- Each row constitutes everything we know about one element of some kind: people in this case

And so, it's time we called at least some of these "things" we've been talking about by their proper "official" names:

- A Table is the whole "rectangular", matrix-like collection of information about similar objects of some kind
- A Field is a column of such a Table
- A Record is a data row of such a Table (i.e., the very first one with column captions doesn't count)

Incidentally, some people even use the words Column in place of Field and Row in place of Record.

Using our newly-acquired lexicon, we can then state, for example, that what's written in a particular cell

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

in the Memo Table that appears in the text above, is a Field Value for the Record that cell belongs to. We can also say that the collection of all Fields of a Table constitutes its Structure.
Finally, a collection of Tables is a Database.

As we have seen above, each Field is dedicated to storing a particular kind of information about the object. One Field might store first names. Another might be dedicated to recording appointment dates. Yet another – prices. Thus, each Field has a preference for particular kinds of data: text strings, dates, numbers, and so on. We could probably store all these pieces of data in textual form – much as we do on paper, but there's a more convenient way: Field Type. The Field that stores first names would be of the String Type. A Date Type Field would be suitable for appointment dates. A Currency Field – for prices. It may not be immediately obvious to you why such "restrictions" on each Field are useful. We'll give you some examples without going into them any further. Having specific Types for Fields ensures that a Field dedicated to storing prices can only contain prices. Otherwise, a price list could end up looking like this: $4.95, $3.25, "cheap", $2.05, "expensive", etc. Enforcing Types makes it easier to sort Records by values of these Fields. It makes it possible to perform some useful operations on them, such as calculating their total, maximum, minimum, and other aggregate values. Almost all Database Systems, Alventis included, offer you a broad range of Field Types, such as String, Integer, Currency, Date, Time, and so on.

Structures of Tables can vary wildly. One Table will contain the Fields we have been listing above: first name, last name, telephone number. Another might have a completely different set: film title, year, director, rating, category. There are virtually no limits to their variations. Each Table is suitable for enumerating a particular type of objects and is structured accordingly. Alventis does not limit the freedom of each Table to have whatever Fields it wants. On the other hand, Alventis does introduce a little bit of order in this chaos. This is accomplished by requiring every Table to have a few "standard" Fields, that we also refer to as System Fields. Everything else is up to the Table's designer (quite possibly you). This dichotomy allows Alventis to always know at least something about each and every Table it deals with, while still permitting the rest of the Table's Structure to vary as necessary.

System Fields. The list of the standard System Fields is quite short:

ID	Auto-Incrementing Integer	This is an integer number that uniquely identifies each Record in a Table. No two Records will ever have the same ID value. This number is automatically assigned to each Record at the moment when the Record is created, and it simply keeps incrementing for each new Record. We intentionally avoid the subject of indexes (indices for purists, but the most widely used spelling is the incorrect one, so we'll stick to that) but for "data-aware readers" among you: the ID is the primary index or key of each Table.
CreationDate	Date/Time	This Field holds the date and time the Record was created. This "timestamp" is recorded with millisecond precision. It could be useful to know when a certain Record first came into existence. It may be even more useful to be able to sort Records by this Field thus being able to list Records in their order of "maturity".

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

LastModificationDate	Date/Time	Another Field of the same Type. This one records the moment of the last time the Record was saved. Note that Alventis uses the timestamp of the local computer on which it is running, not that of the remote Server (in cases of Remote Databases). It is therefore important to ensure that all computers running Alventis have properly-set clocks.
Author	Integer	This Field records the ID of the user who created the Record. Yes, users are also listed in some Table and they too have their unique ID values. It may be useful to simply know who created a certain Record (to thank him/her, obviously). Luckily, under the most common circumstances, Alventis will display the Username of the user in place of his ID. We won't discuss the details here, but Alventis may also use the Author value for other purposes, such as giving access to a particular Record only to its Author. You can learn all about it in the chapter on Alventis Security, if you're fearless enough.
Subject	String	This is a general-purpose String Field that you can use in whatever way you find convenient.
Category	String	Another general-purpose Field – similar to Subject in all respects.

You can record whatever textual information you want in these last two Fields. They default to a maximum Length of 50 characters, but they may have other Lengths in some Tables. If you have Alventis Designer, you can set their Length to whatever you want in the range of 1 to 512 characters. Only String Fields have the Length property which specifies how much text the Field can hold. Each Record will allocate the same amount of space for a String Field regardless of how much text each Record actually ends up containing. Setting the Length to a value that is too high will therefore waste space and lead to a larger Table file on your disk. This is why it makes sense to set String Field's Length to some reasonable value based on what the Field is supposed to hold. If it's a first name Field, these are not likely to exceed, say, 24 characters, so a value in this ballpark would make sense. But we digress. There's one special thing about the Subject and Category Fields that differentiates them from any other String Field you may create or encounter: these two appear in the Search Results grid. As a matter of fact, all System Fields appear there because they are common to all Alventis data Tables. Such being the case, you may want to use the Subject and Category Fields for some of the most recognizable textual information in whatever Table Structures you might create. While it is true that these two fields will always be identified as "Subject" and "Category" in the Search Results grid, you can give them whatever Captions you want on the InfoView forms based on them. For example, a Table of information about cars would likely need such Fields as make and model. Assuming for the moment that these are String Fields, it could be convenient to store the make in the Subject Field and the model name in Category. This way, the Search Results will display the make and model directly in the grid.

Relational Databases. Imagine you have two Tables. The first table is called Countries and – by pure coincidence – holds Records of Countries of the world. Its Structure and some data may look like this:

CountryID	Country	Capital	Population
1	France	Paris	60,000,000
2	Italy	Rome	58,000,000
3	Spain	Madrid	40,000,000

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

This Table may have many more interesting Fields and will most certainly have many more Records for other countries, but the above is sufficient for our little example.

The second Table is called Contacts and it lists Records of people. We've already encountered this Table in earlier examples, but we will now want to record what country each person is from, so it will look as follows:

PersonID	FName	LName	FromCountry
1	Edith	Piaf	France
2	Antonio	Banderas	Spain
3	Federico	Fellini	Italy
4	Jean	Rochefort	France
5	Luchino	Visconti	Italy

As you can see, each Record contains information about a person, including the country he/she is from. Given the fact that we have not one but two Tables, we could, for example take a person's Record, find out from it that person's country, and then go to the other Table and examine that country in more detail. In a certain sense, we have established a relationship between these two Tables. Conceptually, this is exactly how Relational Databases work. One table lists one kind of objects plus a certain value that uniquely identifies some Record from another table. In the above example, this value was the name of the country.

The above two Tables could actually form a correct Relational Database exactly as we have presented them. The presented setup does have a drawback though. Every time we have to enter a new person into the Contacts Table, we have to type his country into the FromCountry String Field. The above was not too bad, but what if you have a long list of Swiss people to enter? The other problem stems from the previous one: what if you misspell a country? Switzerland is a long word, you know. And you can only hope you don't know anybody from Liechtenstein, Tajikistan, and Guinea-Bissau. From the purely spelling perspective, obviously. If you were to misspell one, it would no longer be found in the Countries Table, so the Relational link would be broken. Sure, there are ways to ensure you don't misspell anything, and even let you simply pick an existing country and record its correct name for you, but there's an even simpler way: not record the country name at all. Instead of recording the country name, we can record something else that would let us uniquely identify the country in the Countries Table. Luckily, that Table has just what we're looking for: a unique integer number called CountryID. Our new improved version of the Contacts Table then becomes:

PersonID	FName	LName	FromCountry
1	Edith	Piaf	1
2	Antonio	Banderas	3
3	Federico	Fellini	2
4	Jean	Rochefort	1
5	Luchino	Visconti	2

Note that we can no longer say in all honesty that the last Field holds the "country": "1" is not a country. What we *can* say is that it points to a Record in the Countries Table, and it is that Record that does in fact represent the country. Alventis refers to such Fields as Pointer Type Fields. Internally, they are just integers of course, but Alventis treats them as a special Field Type. Since such Fields are responsible for establishing and maintaining a relationship between two Tables, they certainly deserve some special treatment.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The last version of the Contacts Table certainly has one advantage over its predecessor: you can't misspell a number, so the CountryPtr Field is more reliable. Incidentally, it is also faster and easier to store for the Table. The bad news is that it's just a number, and you may not remember which one to enter for Zimbabwe. The good news is: you don't have to remember anything. Alventis will let you simply select the country you want from a combo box. You may never even find out what the corresponding Country ID value is. Let's take a closer look at that combo box now.

Lookup Combo Box. Its name is its function. It lets you look-up the country you wish in the combo box's dropdown list. When the dropdown is closed, the combo box still quietly performs one very important task: it takes the CountryPtr value recorded in the current Contacts Record, finds the corresponding country in the Countries Table, and displays the country's name.

When we activate/open the dropdown, it lists all countries available in the Countries Table. The main purpose of this is for us to be able to pick the country we want and assign it (well, its ID value) to the current Record's CountryPtr Field. We can do so by either double-clicking the country in the dropdown list or hitting Enter when it is focused.

The above is the most essential "Relational" functionality of the Lookup Combo Box, but it can do a few more useful things. For starters, the dropdown list is not just any list but a whole grid displaying as much (or as little) of the Countries Table as the creator of the InfoView thought necessary. It can theoretically display all Fields from the Countries Table – right in the little dropdown (which is, by the way, resizable). You may be excited to know that this dropdown grid allows editing, so you can even edit the Countries Table directly from it. It may not be the most convenient way, but it can be useful if, say, you notice a small misspelling and want to quickly fix it. If you'd rather edit one of the countries from the dropdown in the convenience of the Countries own InfoView, it's easy to get there. Alt-double-clicking a Record from the dropdown grid will open that Record in its own InfoView. Hitting Alt-Enter will do the same for the focused Record. When the dropdown grid is closed, you can either use one of the above methods to open the Record currently displayed by the combo box, or simply click the little right-pointing arrow at the right of the combo box.

One last aspect of the Lookup Combo Box you should be aware of is that you may encounter more than one level of them. Using the above example, what if the Countries Table also had a Pointer Field, say, pointing to the Continents Table? If this field appeared in the CountryPtr Lookup Combo Box, each country's Record in the dropdown grid would have a similar ContinentPtr Lookup Combo Box right in the grid, so you could drop-down that one too, and so on. There's no limit to how many relational "levels" Tables can have, and Alventis allows all these levels to be implemented in as many cascading Lookup Combo Boxes as necessary. Even if we admit that just 3 levels may get intimidating.

An InfoView form could present the relationship between the Contacts and Countries Tables a little differently. The Lookup "point of view" described above sees Contacts as the "main" Table that accesses the Countries Tables as necessary. This is perhaps the most frequently seen view of their relationship, but not the only possible one. The other possible "point of view" looks at the Countries Tables as the "main" Table. When looking at a Countries Record, we could ask the question: what are the Contacts Records that point to this country? The chapter on Relational Databases Design explains what you must do in Designer to create an InfoView that answers this question. Here, we'll leap straight to the conclusion: the InfoView will look essentially like a Countries Table's InfoView with the addition of a grid of Records from the Contacts Table. As we explore the Countries Table, this "Reverse Lookup" grid will always display just the Contacts Records that point to the current country. It should be noted that the core relationship between the Contacts and Countries Tables has not changed a bit: it's only our "point of view" on that relationship that experienced a kind of "reversal".

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Referential Integrity. Remember our neat little Contacts Table that points to Countries? What do you think would happen to all the Contacts Records from France if we deleted this country from the Countries Table? This would break the Referential Integrity of the link between the Contacts and Countries Tables. Some Contacts Records would suddenly start pointing to no-longer-existing Countries Records. Not good at all. There do exist methods of circumventing this problem by various means. Alventis takes a simple approach: it does not allow you to delete a Record pointed to by some other Record of some relationally-linked Table. Instead, Alventis will inform you of all such Records it finds. If you really absolutely want to delete something (France in the above example), it is your responsibility to "find a new home" for all the French, i.e., re-point each Contacts Record currently pointing to France to some other country (or delete them all – since you are already clearly on the path of destruction). Once this is done, you can delete the Record you want since there are no longer any other Records pointing to it.

In closing, here are some assorted notes on some relevant issues, mostly of interest to readers with some Database background.

We have intentionally not used the master-detail terms since we don't find them particularly intuitive. Same goes for one-to-many and many-to-one relationships.

We haven't said a word about many-to-many relationships and for a simple reason: they don't exist. Period. They are always the result of a premature impression that fails to see that the many-to-many link is actually 2 one-to-many links, with some connecting table in-between, which has simply been overlooked.

The ID field of each table is of course the primary key, and what Alventis refers to as Pointer fields are foreign keys.

Self-referencing tables are not allowed. Nor are circular references.

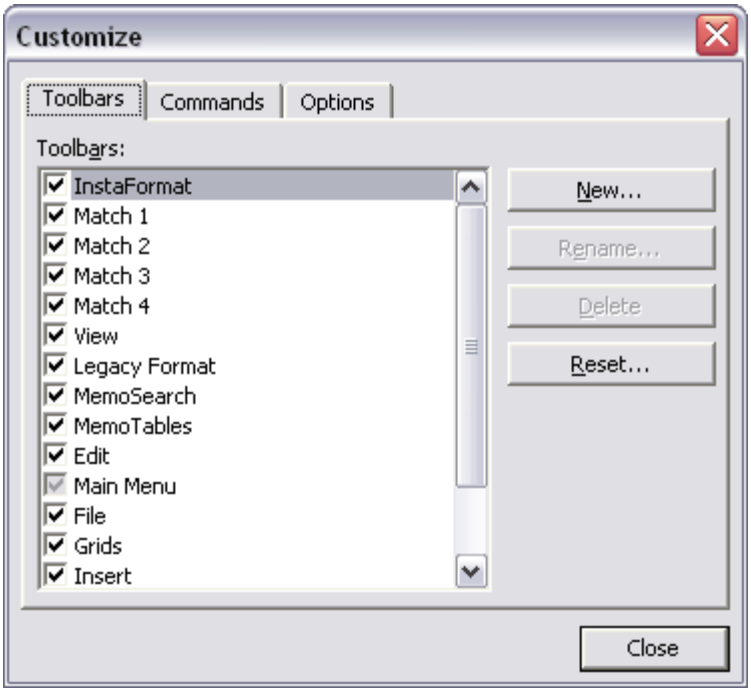
An InfoView may only "implement" the relationship between two tables in either the "Lookup" or the "Reverse Lookup" fashion, but not both at the same time, which wouldn't make any sense anyway.

4.3 Customizing Toolbars and Menus

Both Alventis and Designer allow you full control over their toolbars and menus. Firstly, as we have explained earlier, toolbars and menus are almost entirely mutually-interchangeable. They can contain all the same items, be it buttons or combo boxes. The only difference is the way they present these items to you. You can use either or both, whatever you find convenient. Note that only the main application form's bars/menu are customizable.

You can access the customization pop-up menu by right-clicking anywhere within the menu/bar area. From this menu, you can hide/unhide the bars. You can also access the Customize dialog.

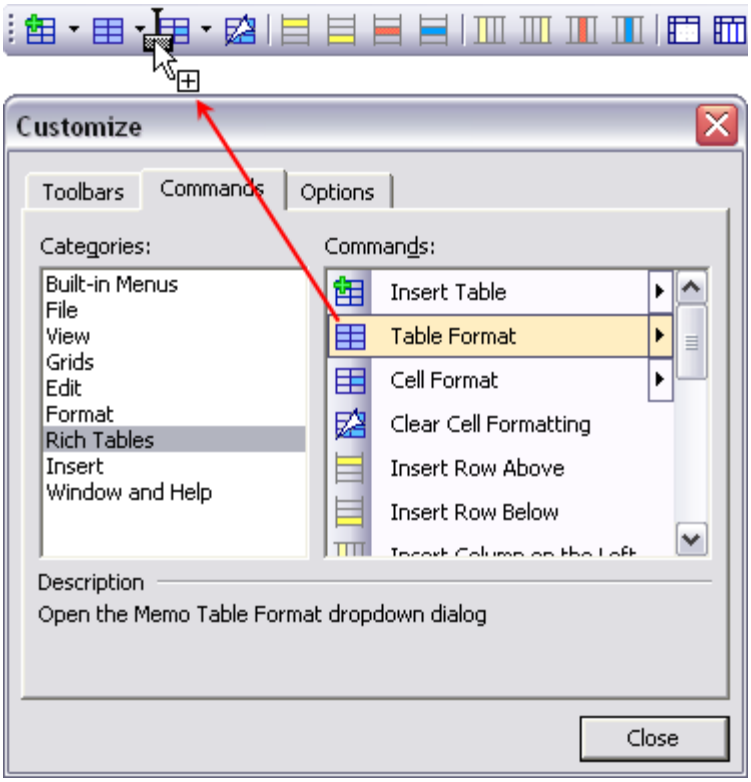
Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Customize dialog has 3 tabs. The Toolbars tab displays a list of existing bars. You can hide/unhide the bars using the corresponding checkboxes. You can create new toolbars, which you can subsequently rename or delete. You can also reset the selected bar/menu to its defaults, which correspond to its "as installed" state. This will discard all your customizations, if any, so be careful.

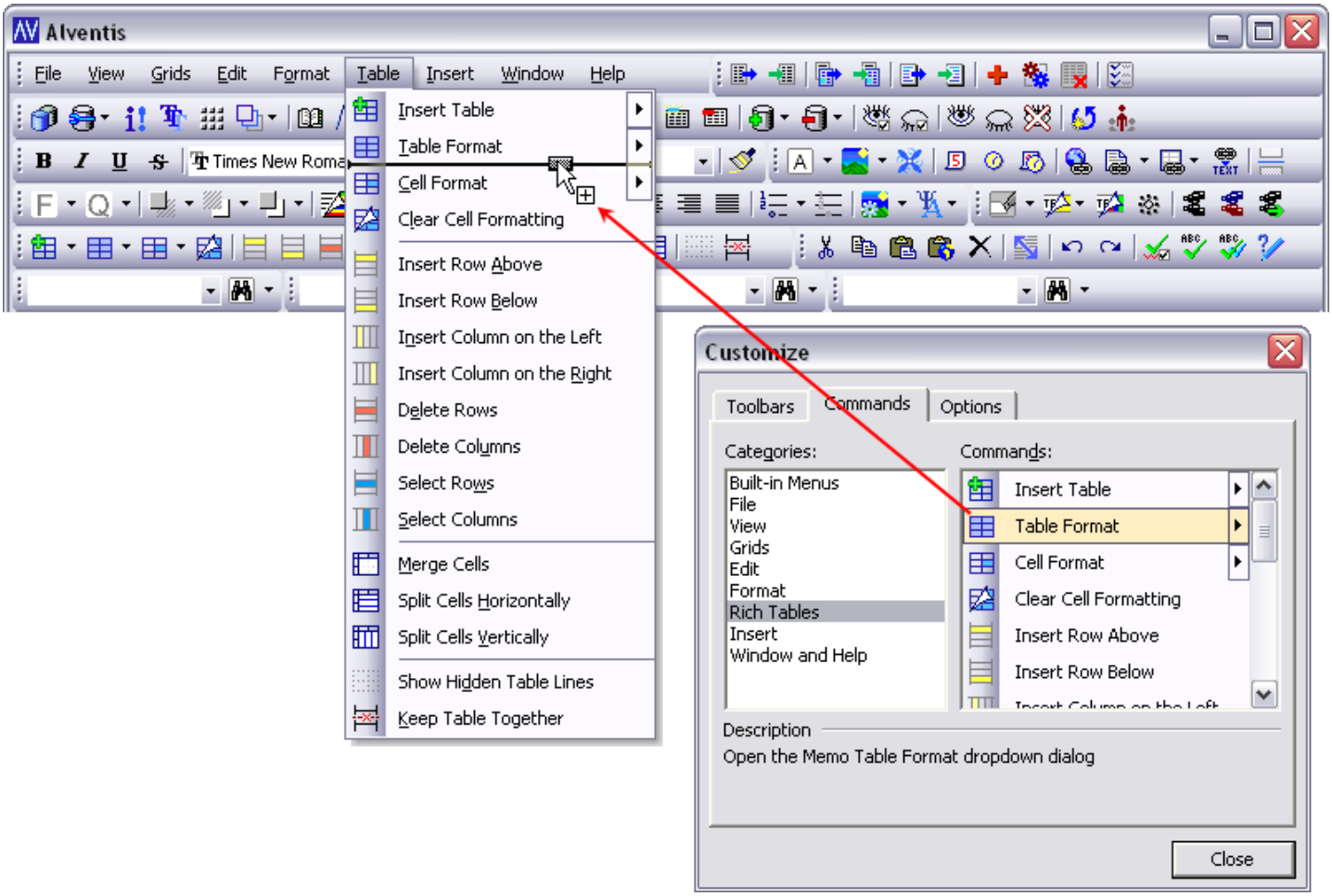
The Commands tab lists all available commands (or items) grouped in Categories seen on the left. Pick a Category to see its commands. You can place any command onto the bar/menu of your choice simply by dragging it in the desired position and dropping it there.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

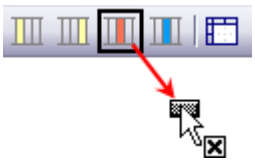


To put an item onto a menu, drag it to the menu caption first – this will open the corresponding menu.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Rearrange existing items using drag-and-drop as well. To remove an item from a bar/menu, drag it "away into nowhere".

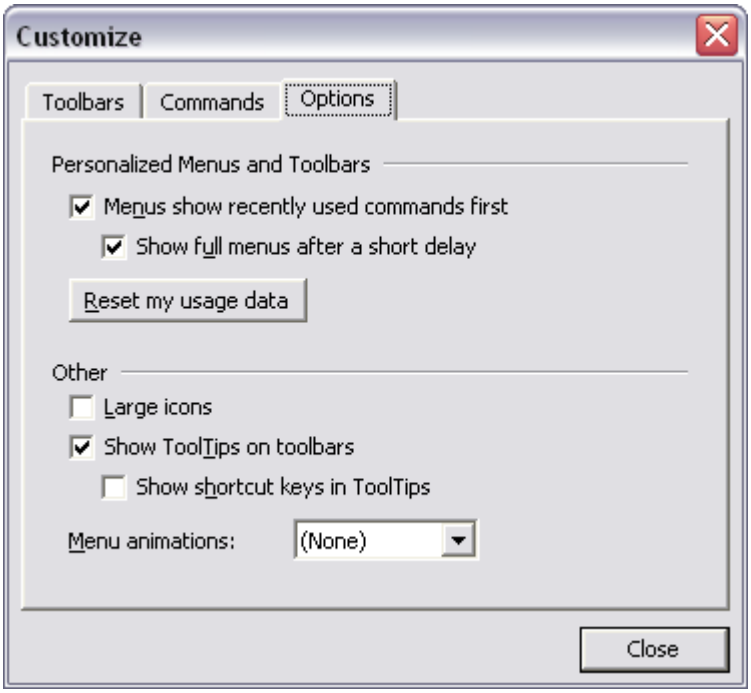


You can create separators between items by dragging one item just a little bit away from the other.

For instance, if [A][B] are adjacent buttons, you'd drag [B] to the right about 1/3 of its width to obtain [A]|[B]. You can't create new items, obviously. You can create new toolbars, but unfortunately, not menus. This is why a "spare" sub-menu called Custom has been pre-created for you. You will find this empty sub-menu at the bottom of the Built-in Menus category.

The Options tab offers a few fairly self-explanatory options. Note that the *Show shortcut keys in ToolTips* option does not work since Alventis and Designer use an alternative mechanism for shortcut processing.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



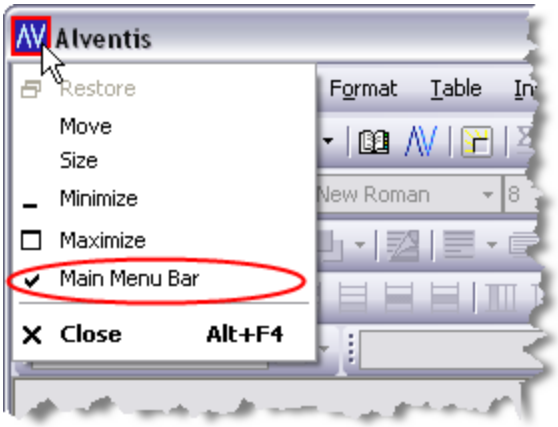
You can move the menu and the toolbars by dragging them with the mouse. Grab them by their grab handle at the very left edge of the bar/menu. Grabbing close to the center of the left edge works best. You can place more than one of them on the same line (including the menu). You can place them either at the top of the application workspace or at the bottom.

InstaButton Multiplicity. InstaButtons deserve a special mention here since this is the type of button you are most likely to want to clone and multiply on the toolbars. Toolbars are more practical than menus for this purpose, so we'll be saying "bars" in the following discussion, but everything applies to menus as well. So, you want 5 Font InstaButtons next to each other? Open the Customize dialog by right-clicking on the bar area and picking it from the pop-up menu.

In the Commands tab, find the right Category (which for Font would likely be Format), then find Font in the Commands list. Drag the Font item and drop it onto the toolbar where you want it. Repeat as many times as you want. Same basic steps apply to all other InstaButtons (and all items in general, of course). You may find it convenient to create a whole new toolbar (see the Toolbars tab and an earlier discussion) to hold all your InstaButtons, but this is entirely up to your taste.

Hiding/Unhiding the Menu. We have seen how you can hide/unhide the bars, but not the menu. This can be done from the application's System menu. Click on the application's icon on the left-hand side of its titlebar (or hit Alt-Space) to invoke this menu.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



The Main Menu Bar command will toggle the menu on/off.

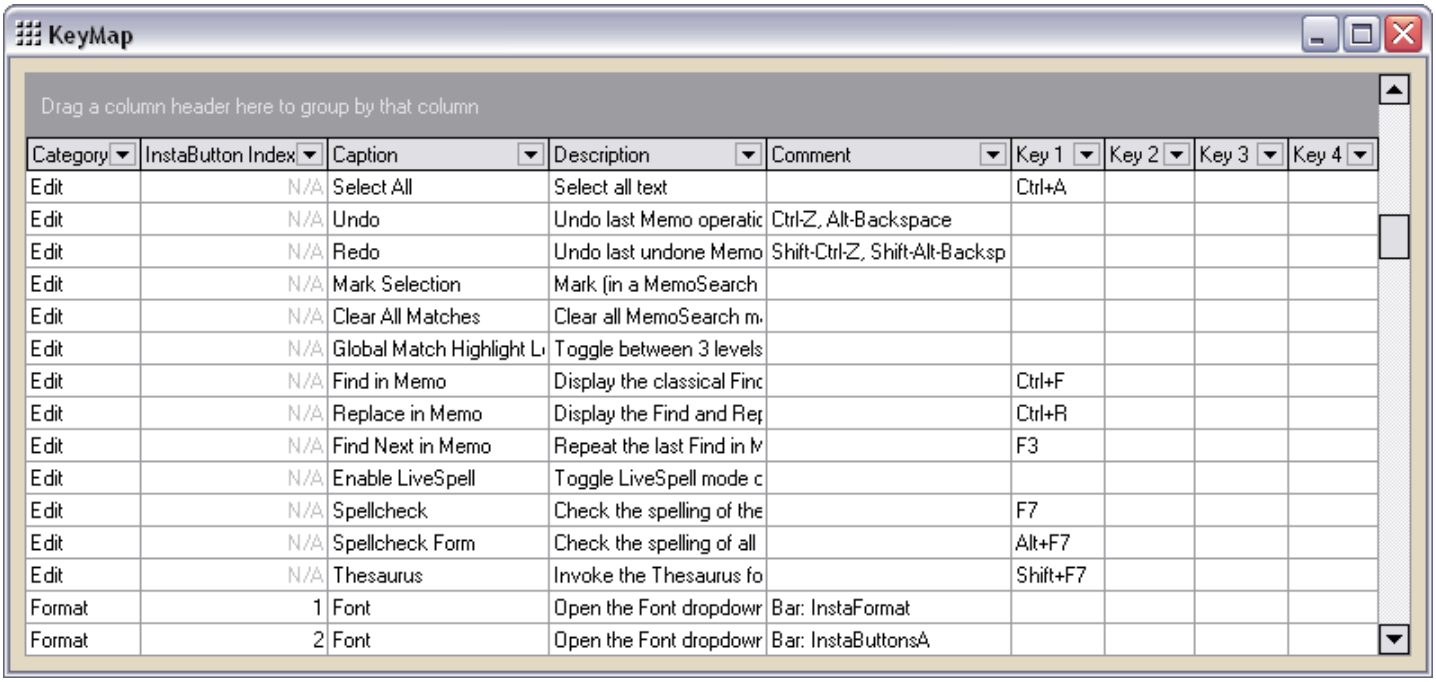
Bar and Menu Style. You can change the visual appearance of all application bars and menus. The 5 possible options are available from the Settingsdialog (the Settings command under the File menu). In the Alventis tradition, your changes take effect immediately, so you can see which look you prefer right away. You can still Cancel your changes though.

KeyMap and Keyboard Shortcuts

In both Alventis and Designer, all keyboard shortcuts are user-definable. You can edit them in the KeyMap form. Note that this is a form and not a dialog: you can leave it open as a "floating reference" or while testing your newly-assigned shortcuts.

All available commands are listed in a single grid. The only columns you can edit are Comment, where you can jot-down a little note to yourself, and the Key 1/2/3/4 columns that specify and allow editing the shortcuts. Yes, there can be as many as 4 shortcuts per command!

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



Category	InstaButton Index	Caption	Description	Comment	Key 1	Key 2	Key 3	Key 4
Edit	N/A	Select All	Select all text		Ctrl+A			
Edit	N/A	Undo	Undo last Memo operatic	Ctrl-Z, Alt-Backspace				
Edit	N/A	Redo	Undo last undone Memo	Shift-Ctrl-Z, Shift-Alt-Backsp				
Edit	N/A	Mark Selection	Mark (in a MemoSearch					
Edit	N/A	Clear All Matches	Clear all MemoSearch m					
Edit	N/A	Global Match Highlight L	Toggle between 3 levels					
Edit	N/A	Find in Memo	Display the classical Finc		Ctrl+F			
Edit	N/A	Replace in Memo	Display the Find and Rep		Ctrl+R			
Edit	N/A	Find Next in Memo	Repeat the last Find in M		F3			
Edit	N/A	Enable LiveSpell	Toggle LiveSpell mode c					
Edit	N/A	Spellcheck	Check the spelling of the		F7			
Edit	N/A	Spellcheck Form	Check the spelling of all		Alt+F7			
Edit	N/A	Thesaurus	Invoke the Thesaurus fo		Shift+F7			
Format	1	Font	Open the Font dropdowr	Bar: InstaFormat				
Format	2	Font	Open the Font dropdowr	Bar: InstaButtonsA				

You will have noticed that the InstaButton Index column displays mysterious numbers for some, but not other commands. The numbers enumerate InstaButtons. A single InstaButton can have multiple independent "incarnations" on the menus and bars. You can assign separate shortcuts to each. Note that it may not be immediately obvious which of the enumerated InstaButtons is which. They should be enumerated in the order in which they were created, but if you don't remember, you may have to resort to trial-and-error approach to identify them.

The Comment column should tell you where the corresponding InstaButton is located (on which toolbar or menu). You can change the default Comment text if you want.

There's another trick to the InstaButton shortcuts. Remember how most of them work (e.g., Font): clicking it applies its stored attribute(s) to the selected item(s) (text, record, item – whatever), and Shift-clicking it sets its attribute(s) to those of the selected item. You can make this work with keyboard shortcuts too: assign two shortcuts to the InstaButton – one *with* and one *without* the Shift key. For example, Ctrl-F11 and Ctrl-Shift-F11. The one invoked with the Shift key depressed will "set" the InstaButton, while the one with no Shift will "use" it. Basically, any InstaButton shortcut with the Shift key will be interpreted as a Shift-click on the InstaButton.

Setting Shortcuts. This part is easy. Get into the cell you want as you would with any grid: focus the desired Key cell and hit Enter or click the already-focused cell with the mouse. Once in the cell, just press the desired shortcut on the keyboard. You'll see the Ctrl-Alt-Shift- key "prefixes" appear and disappear as you press and release them. If you change your mind, hit Escape to abandon the cell. You can clear an existing shortcut by hitting Delete. Duplicate shortcuts are not allowed, and KeyMap will tell you what existing command is using it, if you enter an existing one.

Any shortcut changes you make take effect immediately, so you can test the results right after setting the shortcut (and without leaving the KeyMap form – unless the command in question is only applicable to some particular "place", such as a Memo).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

4.4 International Issues

Alventis supports text in a variety of languages and alphabets. This chapter will explain what is and is not supported, and what pertinent features are at your disposal.

The single most important element of Alventis' multilingual support is Character Set or Charset. This is a property of fonts that determines the subset of characters that are available at any given time. Here's how it works. Imagine you have a Text Box. The Text Box has a Font attribute, and the entire Text Box can only display its contents using a single font. Since Charset is a property or attribute of a font, it follows that the Text Box can only display its contents using one Charset at a time. This Charset (as well as the font) can vary, but the important point and limitation is that only a single Charset can be in effect for any single item at any given time. The only exception to this rule is an Alventis Memo, which can display text in a variety of formats, fonts, and Charsets, but we are only interested in all the other items here.

The above "one Charset" limitation is usually quite tolerable. Let's see what exactly it "means" to a Text Box. It means that once the Text Box's Charset has been set, the Box can only display the available subset of characters. For all textual Charsets (this excludes the special Symbol Charset which has no relevancy to multilingual issues discussed here), the available subset of characters includes all letters of the Latin alphabet (as found in the English language) plus the alphabet of your choice: Greek, Cyrillic, Hebrew, etc. As a result, setting the Charset of your Text Box to Greek will allow you to mix Greek with English (or any other language using the Latin alphabet, but we'll use English in our examples). If the Charset is set to Cyrillic, you can mix Russian and English text, and so on. Basically, the limitation imposed on you by a single Charset limits you to using up to two alphabets at the same time, one of which must be Latin. You cannot mix text from multiple non-Latin alphabets, so you won't be able to put Greek and Russian text in the same Text Box. If this is not on your agenda, you should not encounter any problems.

Items in Alventis and Designer default to the Western Charset, which covers the Latin alphabet and letters with accents and diacritics. If you need to use some other alphabet, say, Cyrillic, you must make some adjustments that vary depending on what it is you are trying to do. We will use Russian/Cyrillic in the following examples, but they apply to any other alphabet.

Russian InfoView. Alright, so you want an entirely Russian InfoView: Russian Labels, Russian text in Text Boxes, etc. Since we're talking about designing an InfoView, this is obviously a task for Designer. Open the InfoView and select the items you want to affect (you can affect them all if you wish by not selecting any items at all: see the Attribute Inheritance section). Change their Font's Charset attribute to Cyrillic. Edit the Labels' Captions as necessary. You're done. To return the Charset to its default "unspecified" value, use the Set Charset to Default button . You can of course have different Charsets for different items, so Labels can be in Russian, and Text Boxes themselves in Greek, for example.

Mixed Contents. Imagine you have a table of films, and you want to record the film's title in its language of origin. This is a tricky proposition since you now have data whose single field's values may use variable alphabets: one record may be listing a Russian film and another – a Greek one. You can't apply the InfoView editing technique we have discussed above because there's no single appropriate Charset for the Title Text Box now. Record Styles to the rescue. In Alventis, set the Record Style of the Russian

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

film record to use the Cyrillic Charset. Similarly, set the Greek film to use the Greek Charset, and so on. As you are navigating from one record to another, Alventis will apply the record's Style's Charset to the entire record, so you'll see its data in the proper alphabet.

Dynamic Charset Adjustments. In theory, the above should be sufficient. You may of course encounter records that have not been assigned a Style at all or one with a wrong Charset. To view and edit such records you can make temporary adjustments to the InfoView directly in Alventis. Click the Select Character Set button to open a list of available Charsets and select the one you want. All Text Boxes and other similar items on the InfoView that are not using some explicitly-specified Charset start using the one you selected. If you want all items to use the new Charset regardless of what their Charset was previously set to, Shift-click the new Charset when selecting it. To return to the original Charset assignments, select the Restore Default option in the Charset list.

The above description implied that some items may have explicitly-specified Charsets. This means that when the InfoView was created in Designer, the Charset of the item was explicitly set to some non-default value.

You can use the Select Character Set button for other Text Boxes or Combo Boxes in Alventis: just focus the Editor of interest and select the new Charset. This may be useful, for example, if you want to perform a search in the UniGrid, and you want to enter the search expression in Russian.
Note that such Charset changes are not permanent and will be forgotten when the relevant form is closed.

Symbol Picker Effect. In Alventis (but not in Designer), whenever you use the Symbol Picker to enter a symbol or some text into a Text Box or a similar Editor, that Editor's Charset is automatically set to that of the symbol you are entering. For example, if you are entering a Greek letter into the UniGrid Search Box using the Symbol Picker, the Search Box's Charset will be set to Greek. If you want to reset the Editor to the original Charset, simply use the Dynamic Charset Adjustments technique described above.

Cut-and-Paste From Web Browsers. We regretfully admit that at present time there appears to be a problem with cutting and pasting international text from many Web browsers into Alventis. This is most likely to affect text in locales other than the default locale used by Windows. The problem is under investigation. Meanwhile, hold on for a twisted workaround that does seem to work. Example: you want to copy some Greek text from a Web page into a Text Box in an InfoView. At some point in time (before the following procedure or after – it doesn't matter), you must ensure the Text Box is set to the Greek Charset. Now, the procedure:

- Copy the text selected in the browser to the clipboard
- Paste it into WordPad (or Microsoft Word)
- Re-select the just-pasted text and copy to the clipboard
- Paste it into any Memo in Alventis (at this point, the text is already OK in the Memo – even if it is not set to the correct Charset)
- Re-select the just-pasted text and copy to the clipboard (yes, again!)
- Paste it into the Text Box

In closing, a little clarification for particularly inquisitive readers: as you may have figured out by now, Alventis does not support Unicode. The reason for this is that Alventis relies on a number of technologies of which only a tiny minority support Unicode. We feel that Alventis' Charset-based implementation provides a sufficiently powerful alternative that will satisfy your multi-lingual needs.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

4.5 Data Table Verification, Repair, and Backup

In a perfect world you would not need this chapter. Alas, in real life, data corruption does happen. We will explain to you the available tools and procedures that should minimize the possibility of you losing your valuable data.

Each table in Alventis is made up of 2 or 3 physical files. If the table is called "Something", you'll find the following files in the directory where the Database is located:

- Something.dat
- Something.idx
- Something.blb

The DAT file holds the table data, except that of BLOB fields (Memos and Graphics/Pictures). The IDX file holds the indices. The BLB file holds the BLOB data mentioned above. It is only present if the table contains at least one Memo or Graphic field, which is why some tables don't have a BLB file.

You may also find files with DBK/IBK/BBK extensions. These are backup files created when you change the table structure in Designer.

When it comes to Alventis data tables of the kind that you use in UniGrids and InfoViews, there's a whole second table that always accompanies the main data table. That second table contains the full-text index for the data table. It follows the same "pattern" as we have described above. It's name is always that of the data table with an ".idw" suffix. If we assume that "Something" table above was one of your data tables, it would then be accompanied by:

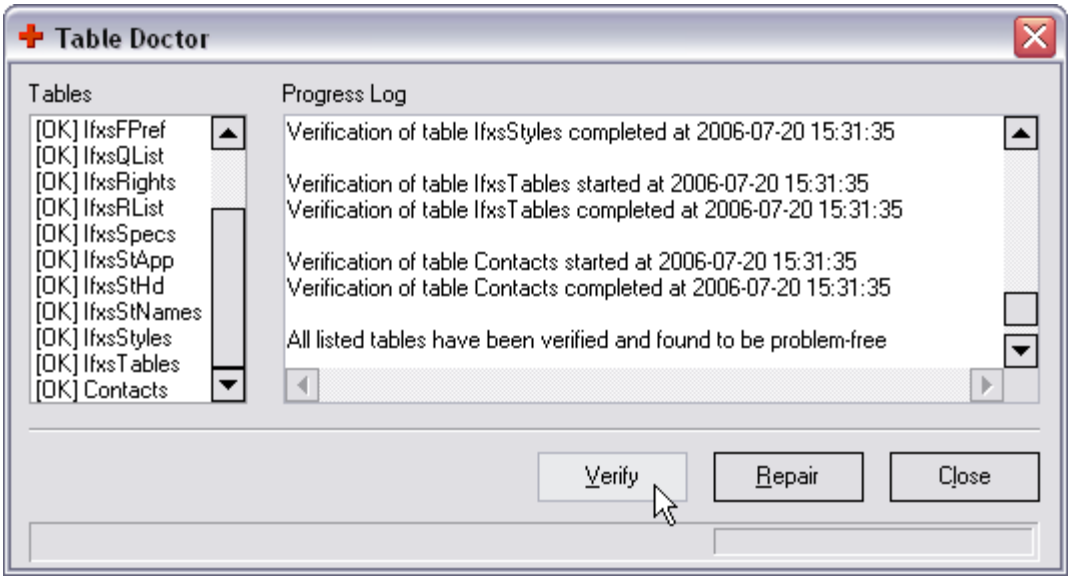
- Something.idw.dat
- Something.idw.idx
- Something.idw.blb

Alventis stores all kinds of information it needs in many other tables. These tables always have names that begin with "Ifxs", e.g., "IfxsTables". Such "system" tables will not have a second full-text index table.

Due to a power outage, a computer crash, or some other sinister event, one or more of the above-mentioned files comprising a table may get corrupted. Some minor corruption can be repaired using tools built into Alventis. Major corruption may not be fixable, so you'd have to restore affected tables from a backup. But let's not be overly pessimistic, so let's talk about repairs first.

Verify / Repair Tables. Clicking the so named button opens the Table Doctor dialog box depicted below:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.



To be able to click this button you must be in the right place (and preferably at the right time – more on that later). Specifically, you must be either in the Tables grid of a UniGrid with some table focused – or in the Database Explorer with some Database focused. When invoked from a UniGrid, the dialog box will offer you to verify and/or repair the selected table and all system tables belonging to the same Database. When invoked from the Database Explorer, all tables of the selected Database will be listed (both data and system ones).

For a table to be verified (or repaired), it must not be already open (neither by your copy of Alventis or Designer, nor by anybody else). If Alventis does not manage to open the table exclusively, verification is immediately abandoned. The Progress Log memo will display a message to that effect:

Operation aborted: Table Something could not be opened exclusively

This in itself does not mean that there's something wrong with the table. It only states that the table is open by someone or something, so it cannot be checked. Which brings us to the question of when is the "right time" to verify/repair tables. It is obviously best to do so when relevant tables can be opened exclusively. Doing so right after you start Alventis may be a good idea. If Alventis re-opens tables on startup, you can either close the corresponding InfoViews manually or simply restart Alventis with the Shift key pressed so that Alventis does not re-open any forms (you do not need to clear Persistence data, just prevent Alventis from using it for this "verification" session). Once you are done with verify/repair operations, you can close Alventis – again holding down the Shift key. This way, the next time it will start and re-open whatever forms you had open – as usual. In a multi-user environment, you must make sure you are the only user accessing the data (not because something can "break" but simply to be able to open the tables exclusively).

The Table Doctor dialog is rather simple. On the left, the tables that will be verified/repaired are listed. On the right is the Progress Log that displays any output messages. Three buttons and a Status bar complete the picture. The Verify button starts verification of all listed tables. Tables that are successfully verified and given a clean bill of health are prefixed with "[OK]" in the list. Tables that could not be verified (typically because they were open) or that did get verified and were found to be defective remain unchanged. If you want to try once more (perhaps after persuading your colleagues to close the table of interest), you can simply click Verify again: only the tables that were not already found to be OK are re-verified. If not all tables are OK, it's a good idea to review the Progress Log to see what's wrong. If you see any messages other than the relatively innocent "could not be opened exclusively",

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

it's time to do some repairs.

Clicking the Repair button launches the repair procedure for all tables not already found to be OK. The basic principle is quite similar to what we described above, except in this case, Alventis will attempt to repair any defects it finds. If you see some messages to the effect that repairs were performed and were successful, it's a very good idea to verify the tables once again (which is why the Repair procedure does not mark the affected tables "[OK]"). On rare occasions Alventis can be overly-optimistic and think it has repaired a table when it has improved it but didn't quite finish the job. If this happens, a second repair will often succeed in fixing the remaining problems. So, occasionally you may need to repeat the Verify/Repair/Verify/Repair cycle. If consecutive attempts to Repair a table produce the same unsatisfactory result, this means the problem is too serious, and Alventis does not and will not manage to fix it. Restoring the table from a backup is then the only solution, but we'll get to that shortly.

The above procedures were verifying and repairing table(s) of one Database. There are some system tables that are not part of any Database though. Alventis verifies them all on startup, every time it is launched. If a problem is found, Alventis will display the Table Doctor dialog on startup offering you an opportunity to repair the corrupted table(s). The procedure is exactly the same as we have described above. Ideally, you should go through the Verify/Repair cycle enough times, until all system tables are OK. If they aren't, closing the dialog in a less-than-perfect state will offer you a choice of closing Alventis (so that you could restore the relevant tables from a backup) or proceeding at your own risk. What is the risk? It's really impossible to tell. The worst that can happen is even worse data corruption leading to further data loss. You should most certainly avoid any data-modification operations (which is rather tricky). Basically, you should fix the problem as soon as possible.

A few notes and facts pertaining to table repairs are in order. A "broken" table may get repaired. This means that you went through the Verify/Repair procedure outlined above, re-verified the table, and that final verification stated the table is now "OK". You must understand though that this only means one thing: the structure of the table no longer has any visible defects. The problem is that nobody can guarantee that the data that was in the table is still intact and what it's supposed to be. For example, imagine your table contained in some string field the name "Smith". Imagine now that some data corruption resulted in one byte of that table changing, so the above field value became "Skith" or some such. The verification procedure will notice the problem, and the repair procedure will most likely clean-up the table structure, but it may not be able to figure out what letter was supposed to be in the 2nd position in that field value, so you might end up with a seemingly healthy table that now contains "Skith". If your data is mission-critical, you may want to, e.g., export it (perhaps from time to time) to text files so that you could perform textual comparisons. If you didn't understand this idea, don't worry about it. This is rather advanced topic that is beyond the scope of this Guide. Besides, data corruption is a fairly rare occurrence.

Index corruption. This paragraph is somewhat advanced, so feel free to skip it. The Repair procedure certainly does its best to repair both data and indices (whatever got corrupted). On rare occasions, the corruption is in the index file, and repairs fail to make any improvements. If you find yourself in that situation, the solution is simple for someone who is comfortable with file management. Close Alventis, delete the broken index file (affected TableName.idx), re-launch Alventis, and try to repair the table again. This time, Alventis should be able to simply rebuild the index from scratch (instead of trying in vain to repair the damage). This works because the index can be "derived" in its entirety from the data file.

Reindex Table. Sometimes, it is not the structure of a table that has problems, but merely its full-text index. The bad news is that there is no way to test for this condition. You will simply notice the

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

consequences. The most common ones are: an error or warning message suggesting you to reindex a table, and UniGrid Search not finding records it should be finding or finding those it shouldn't. The good news is that full-text indexing problems are much less serious than data corruption we have discussed above, and present no risk of any data loss.
When you have reasons (outlined above) to believe that a table may have a problem with its full-text index, all you have to do is rebuild the said index. To do so, select the table of interest in the Tables grid of a UniGrid, and click the Reindex Table button. You get a Proceed/Cancel type of prompt. If you proceed, Alventis will attempt to re-create the full-text index for the selected table. If reindexing fails, you will get a grim message to that effect. Failure almost always means the table could not be opened exclusively (even though the actual error message text may suggest otherwise). If you firmly believe that the relevant table is closed, trying again in several minutes (5 or so) often helps. You can also try closing and reopening Alventis, perhaps holding down the Shift key to prevent the possibility of the relevant table getting automatically reopened.

Technical notes (advanced material). The reindexing operation essentially recreates the entire full-text index from scratch (or more precisely, from the indexed data). This has some important implications. The reindexing process can be quite lengthy for large tables, especially if the tables have Memo fields. Around 1 minute per Megabyte of data file should not be too surprising, so you may want to launch reindexing of large tables at strategically-chosen times of the day. As we said earlier, the full-text index table (confusing but correct terminology: the full-text index is itself stored in a separate table), so, that full-text index table can be completely derived from the data that is being indexed. The full-text index table, as you may remember, is stored in a table that has a name of the type "Something.idw" with extensions dat/idx/blb (3 files). In case the full-text index ("idw") table gets corrupted, you can simply delete it and rebuild it from scratch. After all, this is what is automatically done after an InfoSet Import.

Backup. We have learned that data corruption may occasionally occur and that Alventis may be able to verify and often repair minor corruption. More importantly, we have hopefully learned that not all corruption can be repaired, and even seemingly successful repairs may result in you losing some data. Which brings us to the inevitable subject of backups.

Data backups are the only means of ensuring complete safety of your data. We are not writing a book about backups though, so we'll get straight to the technicalities.

What should be backed up. The very short answer is: data. Everything that looks like data. In the case of Alventis, this means: all files with the following extensions: dat, idx, blb. These files can be found in the following locations:

- The data folder and the "System" subdirectory thereof. The data folder's location varies depending on the version of Windows you are using and on Windows setup. For Windows versions prior to Windows 2000 (i.e., Windows 95/98/ME/NT 4.0), this location is that of the application directory (i.e., the one where you have installed Alventis). For Windows 2000 and newer, this location depends on how Windows was set up, but it will be typically found under "C:\Documents and Settings\All Users\Application Data" or similar. You can locate it by searching your hard disk(s) for the file named "IfxsDBs.dat". You can instruct Alventis and Designer to use a data folder other than the default one described above by launching the applications with the /D command line option, e.g., `Alventis.exe /D="C:\Some Folder Path"`. Paths that contain spaces must be enclosed in double quotes as in the above example.
- Secondly and most importantly: the Database directories. These directories can be anywhere: their locations are specified in the Database Explorer under the "Local" pseudo-server. If relative paths are specified, such paths would be relative to the application directory.

In addition to data, you may also want to backup your custom spelling dictionaries. These would include the files in the Spell subdirectory of the application directory: those with the adu extension. You may

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

also backup Learning.adl and Spell.cfg, even though these 2 are less valuable.

It is of course a good idea to verify all the tables prior to backing them up. A sensible sequence would therefore go something like this: close Designer (if you have it and it was open) and Alventis; re-launch Alventis (so that it gets an opportunity to verify all system tables on startup); verify (and if necessary, repair) all tables of all databases; close Alventis once everything seems in order; perform the backup.

The following notes are rather technical and somewhat advanced, so you can safely ignore this section if you don't fully understand it. From our earlier discussion of repairs and reindexing it should be clear that the truly most important files are data table files (those in the Database directories), of which the dat and blb files are most important since this is where the actual data is stored. The idx file can be rebuilt, but it's obviously better to have a backup copy. The "idw" table is less important in a backup since it can be rebuilt, but it's best to back it up too (if space is not an issue). The system (Ifxs*) tables are for the most part quite important, so you should back them all up. They typically don't take too much space either.

It may be useful to list files you most likely would not want to back up (at least not more than once). The application executables (the files with the exe extension) need not be backed up since you can hopefully re-install the application if they ever got corrupted. Same goes for the User's Guide and the Spelling and Thesaurus dictionaries: files with adm and adt extensions in the Spell subdirectory. None of these files should ever change, so you may want to exclude them from your daily backup routine.

If you ever find yourself restoring data from a backup, you'd first want to make sure all copies of Alventis and/or Designer are closed (and the Server is shut down, if applicable). You would then restore the files you deem necessary (maybe all of them, maybe just one table – this would depend on the situation, so we can't offer you much guidance in this matter). Once this is done, it may be wise to launch Alventis, verify the affected tables, and possibly reindex them (unless you restored both the table and its full-text index).

In closing, we would like to once again remind you of the importance of making backups. Computers crash, hard drives die, power goes out, and in the resulting darkness you step on the CD-R with the only good backup. Make multiple backups of any data you would be sad to lose. On different types of media: other hard drives, diskettes, CD-Rs, Zip disks, Flash cards or drives, etc. Never overwrite your last good backup with a new one. Keep several sets of good backups and rotate them. E.g., if backing up to disks A, B, and C, do so in a round-robin fashion. It may sound tedious, and it probably is, but one day, it may save your data.

4.6 System Capacities

For the most inquisitive among you, following is a list of data-related system capacities as supported by Alventis.

- The maximum number of tables in a database is 4096.
- The maximum number of records in a table is 200 million.
- The maximum record size is 65280 bytes (this does not take BLOB fields into account).
- The maximum number of fields in a table is 1024.
- The maximum number of BLOB fields in a table is 128.
- The maximum size of a BLOB field is 2 gigabytes.
- The maximum length of a string field is 512 bytes.
- The maximum size of the each of the physical files that make up a table (TableName.dat,

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

TableName.idx, and TableName.blb), is 4,000,000,000 bytes.
- The maximum length of a database name is 60 bytes.
- The maximum length of a table name is 60 bytes.
- The maximum length of a table description is 100 bytes.
- The maximum length of a field name is 60 bytes.
- The maximum length of a field description is 100 bytes.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

DBISAM SQL Reference

5 DBISAM SQL Reference

5.1 Overview

The following SQL Reference is provided to you courtesy of Elevate Software, the manufacturer of DBISAM – the database engine used by Alventis. We are including the Reference almost verbatim, so don't be surprised to see numerous and as yet unfamiliar references to DBISAM. It is simply the engine ticking under the hood of Alventis. Note also that all references to "full text indexing" in the SQL Reference have nothing to do with the likewise-named feature of Alventis. Alventis does not use DBISAM's full text indexing (it uses a separate Rubicon engine to achieve a similar effect). We chose to include all information for the sake of completeness – even though you are likely to only ever need a small subset thereof.

DBISAM does not support the complete ANSI SQL-92 specification. Rather, it supports a subset of the specification that includes the most widely used SQL statements for data manipulation and definition, in some cases with DBISAM extensions, as well as some SQL statements that are specific to DBISAM:

SQL Statement	Standard
SELECT	SQL-92 with DBISAM Extensions
INSERT	SQL-92 with DBISAM Extensions
UPDATE	SQL-92 with DBISAM Extensions
DELETE	SQL-92 with DBISAM Extensions
CREATE TABLE	SQL-92 with DBISAM Extensions
ALTER TABLE	SQL-92 with DBISAM Extensions
EMPTY TABLE	DBISAM-specific
OPTIMIZE TABLE	DBISAM-specific
EXPORT TABLE	DBISAM-specific
IMPORT TABLE	DBISAM-specific
REPAIR TABLE	DBISAM-specific
UPGRADE TABLE	DBISAM-specific
DROP TABLE	SQL-92 with DBISAM Extensions
RENAME TABLE	DBISAM-specific
CREATE INDEX	SQL-92 with DBISAM Extensions

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

DROP INDEX	SQL-92 with DBISAM Extensions
START TRANSACTION	SQL-92 with DBISAM Extensions
COMMIT	SQL-92 with DBISAM Extensions
ROLLBACK	SQL-92 with DBISAM Extensions

5.2 Naming Conventions

Introduction

DBISAM requires that certain naming conventions be adhered to when executing SQL. The following rules and naming conventions apply to all supported SQL statements in DBISAM.

Table Names

ANSI-standard SQL specifies that each table name must be a single word comprised of alphanumeric characters and the underscore symbol (_). However, DBISAM's SQL is enhanced to support multi-word table names by enclosing them in double quotes ("") or square brackets ([]):

```
SELECT *
FROM "Customer Data"
```

DBISAM's SQL also supports full file and path specifications in table references for SQL statements being executed within a local session. Table references with path or filename extensions must be enclosed in double quotes ("") or square brackets ([]). For example:

```
SELECT *
FROM "c:\sample\parts"
```

or

```
SELECT *
FROM "parts.dat"
```

Note
It is not recommended that you specify the .dat file name extension in SQL statements for two reasons:

1) First of all, it is possible for the developer to change the default table file extensions for data, index, and BLOB files from the defaults of ".dat", ".idx", and ".blb" to anything that is desired.

2) Using file paths and extensions at all in SQL statements makes the SQL less portable to other database engines or servers.

DBISAM's SQL also supports database name specifications in table references for SQL statements being executed within a remote session. Table references with database must be enclosed in double quotes ("") or square brackets ([]). For example:

```
SELECT *
FROM "\Sample Data\parts"
```

Note
The database name used with remote sessions is not a directory name like it is with local sessions.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Instead, it must be a logical database name that matches that of a database defined on the database server that you are accessing with the SQL statement.

To use an in-memory table in an SQL statement within both local and remote sessions, just prefix the table name with the special "Memory" database name:

```
SELECT *
FROM "\Memory\parts"
```

Column Names

ANSI-standard SQL specifies that each column name be a single word comprised of alphanumeric characters and the underscore symbol (_). However, DBISAM's SQL is enhanced to support multi-word column names. Also, DBISAM's SQL supports multi-word column names and column names that duplicate SQL keywords as long as those column names are enclosed in double quotes ("") or square brackets ([]) or prefaced with an SQL table name or table correlation name. For example, the following column name consists of two words:

```
SELECT E."Emp Id"
FROM employee E
```

In the next example, the column name is the same as the SQL keyword DATE:

```
SELECT weblog.[date]
FROM weblog
```

String Constants

ANSI-standard SQL specifies that string constants be enclosed in single quotes (''), and DBISAM's SQL follows this convention. For example, the following string constant is used in an SQL SELECT WHERE clause:

```
SELECT *
FROM customer
WHERE Company='ABC Widgets'
```

Note
String constants can contain any character in the ANSI character set except for the non-printable characters below character 32 (space). For example, if you wish to embed a carriage-return and line feed in a string constant, you would need to use the following syntax:

```
UPDATE customer SET Notes='ABC Widgets'+
#13+#10+'Located in New York City'
```

The pound sign can be used with the ordinal value of any ANSI character in order to represent that single character as a constant.

Date, Time, TimeStamp, and Number Constants

DBISAM's SQL uses ANSI/ISO date and number formatting for all date, time, timestamp (date/time), and number constants, which is consistent with ANSI-standard SQL except for missing support for date and time interval constants, which are not supported in DBISAM's SQL currently. The formats are as follows:

Constant	Format
Dates	The date format is yyyy-mm-dd where yyyy is the year (4 digits required), mm is the month (leading zero optional), and the day (leading

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

	zero optional).
Times	The time format is hh:mm:ss.zzz am/pm where hh is the hour (leading zero optional), mm is the minutes (leading zero optional), ss is the seconds (leading zero optional), zzz is the milliseconds (leading zero optional), and the am/pm designation for times using the 12-hour clock. The seconds and milliseconds are optional when specifying a time, as is the am/pm designation. If the am/pm designation is omitted, the time is expected to be in 24-hour clock format.
Timestamps (date/time)	The timestamp format is a combination of the date format and the time format with a space in-between the two formats.
Numbers	All numbers are expected to use the period (.) as the decimal separator and no monetary symbols must be used. DBISAM's SQL does not support scientific notation in number constants currently.

All date, time, and timestamp constants must be enclosed in single quotes ('') when specified in an SQL statement. For example:

```
SELECT *
FROM orders
WHERE (saledate <= '1998-01-23')
```

Boolean Constants

The boolean constants TRUE and FALSE can be used for specifying a True or False value. These constants are case-insensitive (True=TRUE). For example:

```
SELECT *
FROM transfers
WHERE (paid = TRUE) AND NOT (incomplete = FALSE)
```

Table Correlation Names

Compliant with ANSI-standard SQL, table correlation names can be used in DBISAM's SQL to explicitly associate a column with the table from which it is derived. This is especially useful when multiple columns of the same name appear in the same query, typically in multi-table queries. A table correlation name is defined by following the table reference in the SQL statement with a unique identifier. This identifier, or table correlation name, can then be used to prefix a column name. The base table name is the default implicit correlation name, irrespective of whether the table name is enclosed in double quotes ("") or square brackets ([]). The base table name is defined as the table name for the DBISAM table not including the full path or any file extensions. For example, the base table name for the physical table "c:\temp\customer.dat" is "customer" as show in this example:

```
SELECT *
FROM "c:\temp\customer.dat"
LEFT OUTER JOIN "c:\temp\orders.dat"
ON (customer.custno = orders.custno)
```

You may also use the physical file name for the table as a table correlation name, although it's not required nor recommended:

```
SELECT *
FROM "customer.dat"
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
LEFT OUTER JOIN "orders.dat"
ON ("customer.dat".custno = "orders.dat".custno)
```

Note
Table correlation names are case-sensitive in any Kylix version of DBISAM. This is due to the fact that table names under Linux are case-sensitive, and since the default table correlation names are based upon the table names the table correlation names must also be case-sensitive.

And finally, you may use a distinctive token as a correlation name (and prefix all column references with the same correlation name):

```
SELECT *
FROM "customer" C
LEFT OUTER JOIN "orders" O
ON (C.custno = O.custno)
```

Column Correlation Names

You can use the AS keyword to assign a correlation name to a column or column expression within a DBISAM SQL SELECT statement, which is compliant with ANSI-standard SQL. Column correlation names can be enclosed in double quotes ("") and can contain embedded spaces. The following example shows how to use the AS keyword to assign a column correlation name:

```
SELECT
customer.company AS "Company Name",
orders.orderno AS "Order #",
sum(items.qty) AS "Total Qty"
FROM customer LEFT OUTER JOIN orders ON customer.custno=orders.custno
LEFT OUTER JOIN items ON orders.orderno=items.orderno
WHERE customer.company LIKE '%Diver%'
GROUP BY 1,2
ORDER BY 1
```

You may also optionally exclude the AS keyword and simply specify the column correlation name directly after the column, as shown here:

```
SELECT
customer.company "Company Name",
orders.orderno "Order #",
sum(items.qty) "Total Qty"
FROM customer LEFT OUTER JOIN orders ON customer.custno=orders.custno
LEFT OUTER JOIN items ON orders.orderno=items.orderno
WHERE customer.company LIKE '%Diver%'
GROUP BY 1,2
ORDER BY 1
```

Embedded Comments

Per ANSI-standard SQL, comments, or remarks, can be embedded in SQL statements to add clarity or explanation. Text is designated as a comment and not treated as SQL by enclosing it within the beginning /* and ending */ comment symbols. The symbols and comments need not be on the same line:

```
/*
  This is a comment
*/
SELECT SUBSTRING(company FROM 1 FOR 4) AS abbrev
FROM customer
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Comments can also be embedded within an SQL statement. This is useful when debugging an SQL statement, such as removing one clause for testing.

```
SELECT company
FROM customer
/* WHERE (state = 'TX') */
ORDER BY company
```

Reserved Words

Below is an alphabetical list of words reserved by DBISAM's SQL. Avoid using these reserved words for the names of metadata objects (tables, columns, and indexes). An exception occurs when reserved words are used as names for metadata objects. If a metadata object must have a reserved word as it name, prevent the error by enclosing the name in double-quotes ("") or square brackets ([]) or by prefixing the reference with the table name (in the case of a column name).

```
ABS
ACOS
ADD
ALL
ALLTRIM
ALTER
AND
AS
ASC
ASCENDING
ASIN
AT
ATAN
ATAN2
AUTOINC
AVG
BETWEEN
BINARY
BIT
BLOB
BLOCK
BOOL
BOOLEAN
BOTH
BY
BYTES
CAST
CEIL
CEILING
CHAR
CHARACTER
CHARCASE
CHARS
COALESCE
COLUMN
COLUMNS
COMMIT
COMPRESS
CONCAT
CONSTRAINT
COS
COT
COUNT
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
CREATE
CURRENT_DATE
CURRENT_GUID
CURRENT_TIME
CURRENT_TIMESTAMP
DAY
DAYOFWEEK
DAYOFYEAR
DAYSFROMMSECS
DECIMAL
DEFAULT
DEGREES
DELETE
DELIMITER
DESC
DESCENDING
DESCRIPTION
DISTINCT
DROP
DUPBYTE
ELSE
EMPTY
ENCRYPTED
ESCAPE
EXCEPT
EXISTS
EXP
EXPORT
EXTRACT
FALSE
FLOAT
FLOOR
FLUSH
FOR
FORCEINDEXREBUILD
FROM
FULL
GRAPHIC
GROUP
GUID
HAVING
HEADERS
HOUR
HOURSFROMMSECS
IDENT_CURRENT
IDENTITY
IF
IFNULL
IMPORT
IN
INCLUDE
INDEX
INNER
INSERT
INT
INTEGER
INTERSECT
INTERVAL
INTO
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

IS
JOIN
KEY
LARGEINT
LAST
LASTAUTOINC
LCASE
LEADING
LEFT
LENGTH
LIKE
LOCALE
LOG
LOG10
LONGVARBINARY
LONGVARCHAR
LOWER
LTRIM
MAJOR
MAX
MAXIMUM
MEMO
MIN
MINIMUM
MINOR
MINSFROMMSECS
MINUTE
MOD
MONEY
MONTH
MSECOND
MSECSFROMMSECS
NOBACKUP
NOCASE
NOCHANGE
NOJOINOPTIMIZE
NONE
NOT
NULL
NUMERIC
OCCURS
ON
OPTIMIZE
OR
ORDER
OUTER
PAGE
PI
POS
POSITION
POWER
PRIMARY
RADIANS
RAND
RANGE
REDEFINE
RENAME
REPAIR
REPEAT

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
REPLACE
RIGHT
ROLLBACK
ROUND
RTRIM
RUNSUM
SECOND
SECSFROMMSECS
SELECT
SET
SIGN
SIN
SIZE
SMALLINT
SPACE
SQRT
START
STDDEV
STOP
SUBSTRING
SUM
TABLE
TAN
TEXT
TEXTOCCURS
TEXTSEARCH
THEN
TIME
TIMESTAMP
TO
TOP
TRAILBYTE
TRAILING
TRANSACTION
TRIM
TRUE
TRUNC
TRUNCATE
UCASE
UNION
UNIQUE
UPDATE
UPGRADE
UPPER
USER
VALUES
VARBINARY
VARBYTES
VARCHAR
VERIFY
VERSION
WEEK
WHERE
WITH
WORD
WORDS
WORK
YEAR
YEARSFROMMSECS
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The following are operators used in DBISAM's SQL. Avoid using these characters in the names of metadata objects:

```
|
+
-
*
/
<>
<
>
.
;
,
=
<=
>=
(
)
[
]
#
```

5.3 Unsupported SQL

The following ANSI-standard SQL-92 language elements are not used in DBISAM's SQL:

ALLOCATE CURSOR (Command)
ALLOCATE DESCRIPTOR (Command)
ALTER DOMAIN (Command)
CHECK (Constraint)
CLOSE (Command)
CONNECT (Command)
CONVERT (Function)
CORRESPONDING BY (Expression)
CREATE ASSERTION (Command)
CREATE CHARACTER SET (Command)
CREATE COLLATION (Command)
CREATE DOMAIN (Command)
CREATE SCHEMA (Command)
CREATE TRANSLATION (Command)
CREATE VIEW (Command)
CROSS JOIN (Relational operator)
DEALLOCATE DESCRIPTOR (Command)
DEALLOCATE PREPARE (Command)
DECLARE CURSOR (Command)
DECLARE LOCAL TEMPORARY TABLE (Command)
DESCRIBE (Command)
DISCONNECT (Command)
DROP ASSERTION (Command)
DROP CHARACTER SET (Command)

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

DROP COLLATION (Command)
DROP DOMAIN (Command)
DROP SCHEMA (Command)
DROP TRANSLATION (Command)
DROP VIEW (Command)
EXECUTE (Command)
EXECUTE IMMEDIATE (Command)
FETCH (Command)
FOREIGN KEY (Constraint)
GET DESCRIPTOR (Command)
GET DIAGNOSTICS (Command)
GRANT (Command)
MATCH (Predicate)
NATURAL (Relational operator)
NULLIF (Expression)
OPEN (Command)
OVERLAPS (Predicate)
PREPARE (Command)
REFERENCES (Constraint)
REVOKE (Command)
SET CATALOG (Command)
SET CONNECTION (Command)
SET CONSTRAINTS MODE (Command)
SET DESCRIPTOR (Command)
SET NAMES (Command)
SET SCHEMA (Command)
SET SESSION AUTHORIZATION (Command)
SET TIME ZONE (Command)
SET TRANSACTION (Command)
TRANSLATE (Function)
USING (Relational operator)

5.4 Operators

Introduction

DBISAM allows comparison operators, extended comparison operators, arithmetic operators, string operators, date, time, and timestamp operators, and logical operators in SQL statements. These operators are detailed below.

Comparison Operators

Use comparison operators to perform comparisons on data in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following comparison operators:

Operator	Description
<	Determines if a value is less than another value.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

>	Determines if a value is greater than another value.
=	Determines if a value is equal to another value.
<>	Determines if a value is not equal to another value.
>=	Determines if a value is greater than or equal to another value.
<=	Determines if a value is less than or equal to another value.

Use comparison operators to compare two like values. Values compared can be: column values, literals, or calculations. The result of the comparison is a boolean value that is used in contexts like a WHERE clause to determine on a row-by-row basis whether a row meets the filtering criteria. The following example uses the >= comparison operator to show only the orders where the ItemsTotal column is greater than or equal to 1000:

```
SELECT *
FROM Orders
WHERE (ItemsTotal >= 1000)
```

Comparisons must be between two values of the same or a compatible data type. The result of a comparison operation can be modified by a logical operator, such as NOT. The following example uses the >= comparison operator and the logical NOT operator to show only the orders where the ItemsTotal column is not greater than or equal to 1000:

```
SELECT *
FROM Orders
WHERE NOT (ItemsTotal >= 1000)
```

Note
Comparison operators can only be used in a WHERE or HAVING clause, or in the ON clause of a join - they cannot be used in the SELECT clause. The only exception to this would be within the first argument to the IF() function, which allows comparison expressions for performing IF...ELSE boolean logic.

Extended Comparison Operators

Use extended comparison operators to perform comparisons on data in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM supports the following extended comparison operators:

Operator	Description
[NOT] BETWEEN	Compares a value to a range formed by two values.
[NOT] IN	Determines whether a value exists in a list of values.
[NOT] LIKE	Compares, in part or in whole, one value with another.
IS [NOT] NULL	Compares a value with an empty, or NULL, value.

BETWEEN Extended Comparison Operator

The BETWEEN extended comparison operator determines whether a value falls inside a range. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
value1 [NOT] BETWEEN value2 AND value3
```

Use the BETWEEN extended comparison operator to compare a value to a value range. If the value is greater than or equal to the low end of the range and less than or equal to the high end of the range, BETWEEN returns a TRUE value. If the value is less than the low end value or greater than the high end value, BETWEEN returns a FALSE value. For example, the expression below returns a FALSE value because 10 is not between 1 and 5:

```
10 BETWEEN 1 AND 5
```

Use NOT to return the converse of a BETWEEN comparison. For example, the expression below returns a TRUE value:

```
10 NOT BETWEEN 1 AND 5
```

BETWEEN can be used with all non-BLOB data types, but all values compared must be of the same or a compatible data type. The left-side and right-side values used in a BETWEEN comparison may be columns, literals, or calculated values. The following example returns all orders where the SaleDate column is between January 1, 1998 and December 31, 1998:

```
SELECT SaleDate
FROM Orders
WHERE (SaleDate BETWEEN '1998-01-01' AND '1998-12-31')
```

BETWEEN is useful when filtering to retrieve rows with contiguous values that fall within the specified range. For filtering to retrieve rows with noncontiguous values, use the IN extended comparison operator.

IN Extended Comparison Operator

The IN extended comparison operator indicates whether a value exists in a set of values. The syntax is as follows:

```
value [NOT] IN (value_set)
```

Use the IN extended comparison operator to filter a table based on the existence of a column value in a specified set of comparison values. The set of comparison values can be a comma-separated list of column names, literals, or calculated values. The following example returns all customers where the State column is either 'CA' or 'HI':

```
SELECT c.Company, c.State
FROM Customer c
WHERE (c.State IN ('CA', 'HI'))
```

The value to compare with the values set can be any or a combination of a column value, a literal value, or a calculated value. Use NOT to return the converse of an IN comparison. IN can be used with all non-BLOB data types, but all values compared must be of the same or a compatible data type.

IN is useful when filtering to retrieve rows with noncontiguous values. For filtering to retrieve rows with contiguous values that fall within a specified range, use the BETWEEN extended comparison operator.

LIKE Extended Comparison Operator

The LIKE extended comparison operator indicates the similarity of one value as compared to another. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
value [NOT] LIKE [substitution_char] comparison_value
[substitution_char] ESCAPE escape_char
```

Use the LIKE extended comparison operator to filter a table based on the similarity of a column value to a comparison value. Use of substitution characters allows the comparison to be based on the whole column value or just a portion. The following example returns all customers where the Company column is equal to 'Adventure Undersea':

```
SELECT *
FROM Customer
WHERE (Company LIKE 'Adventure Undersea')
```

The wildcard substitution character (%) may be used in the comparison to represent an unknown number of characters. LIKE returns a TRUE when the portion of the column value matches that portion of the comparison value not corresponding to the position of the wildcard character. The wildcard character can appear at the beginning, middle, or end of the comparison value (or multiple combinations of these positions). The following example retrieves rows where the column value begins with 'A' and is followed by any number of any characters. Matching values could include 'Action Club' and 'Adventure Undersea', but not 'Blue Sports':

```
SELECT *
FROM Customer
WHERE (Company LIKE 'A%')
```

The single-character substitution character (_) may be used in the comparison to represent a single character. LIKE returns a TRUE when the portion of the column value matches that portion of the comparison value not corresponding to the position of the single-character substitution character. The single-character substitution character can appear at the beginning, middle, or end of the comparison value (or multiple combinations of these positions). Use one single-character substitution character for each character to be wild in the filter pattern. The following example retrieves rows where the column value begins with 'b' ends with 'n', with one character of any value between. Matching values could include 'bin' and 'ban', but not 'barn':

```
SELECT Words
FROM Dictionary
WHERE (Words LIKE 'b_n')
```

The ESCAPE keyword can be used after the comparison to represent an escape character in the comparison value. When an escape character is found in the comparison value, DBISAM will treat the next character after the escape character as a literal and not a wildcard character. This allows for the use of the special wildcard characters as literal search characters in the comparison value. For example, the following example retrieves rows where the column value contains the string constant '10%':

```
SELECT ID, Description
FROM Items
WHERE (Description LIKE '%10\%%') ESCAPE '\'
```

Use NOT to return the converse of a LIKE comparison. LIKE can be used only with string or compatible data types such as memo columns. The comparison performed by the LIKE extended comparison operator is always case-sensitive.

IS NULL Extended Comparison Operator

The IS NULL extended comparison operator indicates whether a column contains a NULL value. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
column_reference IS [NOT] NULL
```

Use the IS NULL extended comparison operator to filter a table based on the specified column containing a NULL (empty) value. The following example returns all customers where the InvoiceDate column is null:

```
SELECT *
FROM Customer
WHERE (InvoiceDate IS NULL)
```

Use NOT to return the converse of a IS NULL comparison.

Note
For a numeric column, a zero value is not the same as a NULL value.

Value Operators

Use value operators to return specific values based upon other expressions in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM supports the following value operators:

Operator	Description
CASE	Evaluates a series of boolean expressions and returns the matching result value.

CASE Value Operator

The CASE value operator can be used with two different syntaxes, one being the normal syntax while the other being a shorthand syntax. The normal syntax is used to evaluate a series of boolean expressions and return the matching result value for the first boolean expression that returns True, and is as follows:

```
CASE
WHEN boolean expression THEN value
[WHEN boolean expression THEN value]
[ELSE] value
END
```

The following is an example of the normal CASE syntax. It translate a credit card type into a more verbose description:

```
SELECT CardType,
CASE
WHEN Upper(CardType)='A' THEN 'American Express'
WHEN Upper(CardType)='M' THEN 'Mastercard'
WHEN Upper(CardType)='V' THEN 'Visa'
WHEN Upper(CardType)='D' THEN 'Diners Club'
END AS CardDesc,
SUM(SalesAmount) AS TotalSales
FROM Transactions
GROUP BY CardType
ORDER BY TotalSales DESC
```

The shorthand syntax is as follows:

```
CASE expression
WHEN expression THEN value
[WHEN expression THEN value]
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
[ELSE] value
END
```

The primary difference between the shorthand syntax and the normal syntax is the inclusion of the expression directly after the CASE operator itself. It is used as the comparison value for every WHEN expression. All WHEN expressions must be type-compatible with this expression and can be any type, unlike the normal syntax which requires boolean expressions. The rest of the shorthand syntax is the same as the normal syntax.

The following is the above credit card type example using the shorthand syntax:

```
SELECT CardType,
CASE Upper(CardType)
WHEN 'A' THEN 'American Express'
WHEN 'M' THEN 'Mastercard'
WHEN 'V' THEN 'Visa'
WHEN 'D' THEN 'Diners Club'
END AS CardDesc,
SUM(SalesAmount) AS TotalSales
FROM Transactions
GROUP BY CardType
ORDER BY TotalSales DESC
```

Arithmetic Operators

Use arithmetic operators to perform arithmetic calculations on data in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following arithmetic operators:

Operator	Description
+	Add two numeric values together.
-	Subtract one numeric value from another numeric value.
*	Multiply one numeric value by another numeric value.
/	Divide one numeric value by another numeric value.
MOD	Returns the modulus of the two integer arguments as an integer

Calculations can be performed wherever non-aggregated data values are allowed, such as in a SELECT or WHERE clause. In the following example, a column value is multiplied by a numeric literal:

```
SELECT (itemstotal * 0.0825) AS Tax
FROM orders
```

Arithmetic calculations are performed in the normal order of precedence: multiplication, division, modulus, addition, and then subtraction. To cause a calculation to be performed out of the normal order of precedence, use parentheses around the operation to be performed first. In the next example, the addition is performed before the multiplication:

```
SELECT (n.numbers * (n.multiple + 1)) AS Result
FROM numbertable n
```

Arithmetic operators operate only on numeric values.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

String Operators

Use string operators to perform string concatenation on character data in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following string operators:

Operator	Description
+	Concatenate two string values together.
||	Concatenate two string values together.

String operations can be performed wherever non-aggregated data values are allowed, such as in a SELECT or WHERE clause. In the following example, a column value is concatenated with a second column value to provide a new calculated column in the query result set:

```
SELECT (LastName + ', ' + FirstName) AS FullName
FROM Employee
```

String operators operate only on string values or memo columns.

Date, Time, and Timestamp Operators

Use date, time, and timestamp operators to perform date, time, and timestamp calculations in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following date, time, and timestamp operators:

Operator	Description
+	Adding days or milliseconds to date, time, or timestamp values.
-	Subtracting days or milliseconds from date, time, or timestamp values, or subtracting two date, time, or timestamp values to get the difference in days or milliseconds.

The rules for adding or subtracting dates, times, and timestamps in conjunction with integers are as follows:

Adding an integer to a date is equivalent to adding days to the date

Adding an integer to a time is equivalent to adding milliseconds to the time (be careful of wraparound since a time value is equal to the number of milliseconds elapsed since the beginning of the current day)

Adding an integer to a timestamp is equivalent to adding milliseconds to the time portion of the timestamp (any milliseconds beyond the number of milliseconds in a day will result in an increment of the day value in the timestamp by 1)

Subtracting an integer from a date is equivalent to subtracting days from the date

Subtracting an integer from a time is equivalent to subtracting milliseconds from the time (be careful of going below 0, which will be ignored)

Subtracting an integer from a timestamp is equivalent to subtracting milliseconds from the time portion

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

of the timestamp (any milliseconds less than 0 for the time portion will result in a decrement of the day value in the timestamp by 1)

Subtracting a date value from another date value will result in the number of days between the two dates (be sure to use the ABS() function to ensure a positive value if the second value is larger than the first)

Subtracting a time value from another time value will result in the number of milliseconds between the two times (be sure to use the ABS() function to ensure a positive value if the second value is larger than the first)

Subtracting a date value from a timestamp value will result in the number of milliseconds between the timestamp and the date (be sure to use the ABS() function to ensure a positive value if the second value is larger than the first)

Subtracting a timestamp value from a timestamp value will result in the number of milliseconds between the timestamp and the other timestamp (be sure to use the ABS() function to ensure a positive value if the second value is larger than the first)

The following example shows how you would add 30 days to a date to get an invoice due date for an invoice in a SELECT SQL statement:

```
SELECT InvoiceDate, (InvoiceDate + 30) AS DueDate, BalanceDue
FROM Invoices
WHERE InvoiceDate BETWEEN '1999-01-01' AND '1999-01-31'
```

Date, time, and timestamp operators operate only on date, time, or timestamp values in conjunction with integer values.

Logical Operators

Use logical operators to perform Boolean logic between different predicates (conditions) in an SQL WHERE clause. DBISAM's SQL supports the following logical operators:

Operator	Description
OR	OR two boolean values together.
AND	AND two boolean values together.
NOT	NOT a boolean value.

This allows the source table(s) to be filtered based on multiple conditions. Logical operators compare the boolean result of two predicate comparisons, each producing a boolean result. If OR is used, either of the two predicate comparisons can result in a TRUE value for the whole expression to evaluate to TRUE. If AND is used, both predicate comparisons must evaluate to TRUE for the whole expression to be TRUE; if either is FALSE, the whole is FALSE. In the following example, if at least one of the two predicate comparisons is TRUE, the row will be included in the query result set:

```
SELECT *
FROM Reservations
WHERE ((ReservationDate < '1998-01-31') OR (Paid = TRUE))
```

Logical operator comparisons are performed in the order of OR and then AND. To perform a comparison

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

out of the normal order of precedence, use parentheses around the comparison to be performed first. The SELECT statement below retrieves all rows where the Shape column is 'Round' and the Color is 'Blue'. It also returns those rows where the Color column is 'Red', regardless of the value in the Shape column (such as 'Triangle'). It would not return rows where the Shape is 'Round' and the Color is anything but 'Blue' or where the Color is 'Blue' and the Shape is anything but 'Round':

```
SELECT Shape, Color, Cost
FROM Objects
WHERE ((Shape = 'Round') AND (Color = 'Blue')) OR (Color = 'Red')
```

Without the parentheses, the default order of precedence is used and the logic changes. The next example, a variation on the above statement, would return rows where the Shape is 'Square' and the Color is 'Blue'. It would also return rows where the Shape is 'Square' and the Color is 'Red'. But unlike the preceding statement, it would not return rows where the Color is 'Red' and the Shape is 'Triangle':

```
SELECT Shape, Color, Cost
FROM Objects
WHERE Shape = 'Round' AND Color = 'Blue' OR Color = 'Red'
```

Use the NOT operator to negate the boolean result of a comparison. In the following example, only those rows where the Paid column contains a FALSE value are retrieved:

```
SELECT *
FROM reservations
WHERE (NOT (Paid = TRUE))
```

5.5 String Functions

Use string functions to manipulate string values in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following string functions:

Function	Description
LOWER or LCASE	Forces a string to lowercase.
UPPER or UCASE	Forces a string to uppercase.
LENGTH	Returns the length of a string value.
SUBSTRING	Extracts a portion of a string value.
LEFT	Extracts a certain number of characters from the left side of a string value.
RIGHT	Extracts a certain number of characters from the right side of a string value.
TRIM	Removes repetitions of a specified character from the left, right, or both sides of a string.
LTRIM	Removes any leading space characters from a string.
RTRIM	Removes any trailing space characters from a string.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

POS or POSITION	Finds the position of one string value within another string value.
OCCURS	Finds the number of times one string value is present within another string value.
REPLACE	Replaces all occurrences of one string value within a string by a new string value.
REPEAT	Repeats a string value a specified number of times.
CONCAT	Concatenates two string values together.

LOWER or LCASE Function

The LOWER or LCASE function converts all characters in a string value to lowercase. The syntax is as follows:

```
LOWER(column_reference or string constant)
LCASE(column_reference or string constant)
```

In the following example, the values in the NAME column appear all in lowercase:

```
SELECT LOWER(Name)
FROM Country
```

The LOWER or LCASE function can be used in WHERE clause string comparisons to cause a case-insensitive comparison. Apply LOWER or LCASE to the values on both sides of the comparison operator (if one of the comparison values is a literal, simply enter it all in lower case).

```
SELECT *
FROM Names
WHERE LOWER(Lastname) = 'smith'
```

LOWER or LCASE can only be used with string or memo columns or constants.

UPPER or UCASE Function

The UPPER or UCASE function converts all characters in a string value to uppercase. The syntax is as follows:

```
UPPER(column_reference or string constant)
UCASE(column_reference or string constant)
```

Use UPPER or UCASE to convert all of the characters in a table column or character literal to uppercase. In the following example, the values in the NAME column are treated as all in uppercase. Because the same conversion is applied to both the filter column and comparison value in the WHERE clause, the filtering is effectively case-insensitive:

```
SELECT Name, Capital, Continent
FROM Country
WHERE UPPER(Name) LIKE UPPER('PE%')
```

UPPER can only be used with string or memo columns or constants.

LENGTH Function

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The LENGTH function returns the length of a string value as an integer value. The syntax is as follows:

```
LENGTH(column_reference or string constant)
```

In the following example, the length of the values in the Notes column are returned as part of the SELECT statement:

```
SELECT Notes, LENGTH(Notes) AS "Num Chars"
FROM Biolife
```

LENGTH can only be used with string or memo columns or constants.

SUBSTRING Function

The SUBSTRING function extracts a substring from a string. The syntax is as follows:

```
SUBSTRING(column_reference or string constant
          FROM start_index [FOR length])
SUBSTRING(column_reference or string constant,
          start_index[,length])
```

The second FROM parameter is the character position at which the extracted substring starts within the original string. The index for the FROM parameter is based on the first character in the source value being 1.

The FOR parameter is optional, and specifies the length of the extracted substring. If the FOR parameter is omitted, the substring goes from the position specified by the FROM parameter to the end of the string.

In the following example, the SUBSTRING function is applied to the literal string 'ABCDE' and returns the value 'BCD':

```
SELECT SUBSTRING('ABCDE' FROM 2 FOR 3) AS Sub
FROM Country
```

In the following example, only the second and subsequent characters of the NAME column are retrieved:

```
SELECT SUBSTRING(Name FROM 2)
FROM Country
```

SUBSTRING can only be used with string or memo columns or constants.

LEFT Function

The LEFT function extracts a certain number of characters from the left side of a string. The syntax is as follows:

```
LEFT(column_reference or string constant FOR length)
LEFT(column_reference or string constant,length)
```

The FOR parameter specifies the length of the extracted substring.

In the following example, the LEFT function is applied to the literal string 'ABCDE' and returns the value 'ABC':

```
SELECT LEFT('ABCDE' FOR 3) AS Sub
FROM Country
```

LEFT can only be used with string or memo columns or constants.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

RIGHT Function

The RIGHT function extracts a certain number of characters from the right side of a string. The syntax is as follows:

```
RIGHT(column_reference or string constant FOR length)
RIGHT(column_reference or string constant,length)
```

The FOR parameter specifies the length of the extracted substring.

In the following example, the RIGHT function is applied to the literal string 'ABCDE' and returns the value 'DE':

```
SELECT RIGHT('ABCDE' FOR 2) AS Sub
FROM Country
```

RIGHT can only be used with string or memo columns or constants.

TRIM Function

The TRIM function removes the trailing or leading character, or both, from a string. The syntax is as follows:

```
TRIM([LEADING|TRAILING|BOTH] trimmed_char
     FROM column_reference or string constant)
TRIM([LEADING|TRAILING|BOTH] trimmed_char,
     column_reference or string constant)
```

The first parameter indicates the position of the character to be deleted, and has one of the following values:

Keyword	Description
LEADING	Deletes the character at the left end of the string.
TRAILING	Deletes the character at the right end of the string.
BOTH	Deletes the character at both ends of the string.

The trimmed character parameter specifies the character to be deleted. Case-sensitivity is applied for this parameter. To make TRIM case-insensitive, use the UPPER or UCASE function on the column reference or string constant.

The FROM parameter specifies the column or constant from which to delete the character. The column reference for the FROM parameter can be a string column or a string constant.

The following are examples of using the TRIM function:

`TRIM(LEADING '_' FROM '_ABC_')` will return 'ABC_'
`TRIM(TRAILING '_' FROM '_ABC_')` will return '_ABC'
`TRIM(BOTH '_' FROM '_ABC_')` will return 'ABC'
`TRIM(BOTH 'A' FROM 'ABC')` will return 'BC'

TRIM can only be used with string or memo columns or constants.

LTRIM Function

The LTRIM function removes any leading spaces from a string. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
LTRIM(column_reference or string constant)
```

The first and only parameter specifies the column or constant from which to delete the leading spaces, if any are present. The following is an example of using the LTRIM function:

`LTRIM('   ABC')` will return 'ABC'

LTRIM can only be used with string or memo columns or constants.

RTRIM Function

The RTRIM function removes any trailing spaces from a string. The syntax is as follows:

```
RTRIM(column_reference or string constant)
```

The first and only parameter specifies the column or constant from which to delete the trailing spaces, if any are present. The following is an example of using the RTRIM function:

`RTRIM('ABC   ')` will return 'ABC'

RTRIM can only be used with string or memo columns or constants.

POS or POSITION Function

The POS or POSITION function returns the position of one string within another string. The syntax is as follows:

```
POS(string constant IN column_reference or string constant)
POSITION(string constant IN column_reference or string constant)
POS(string constant,column_reference or string constant)
POSITION(string constant,column_reference or string constant)
```

If the search string is not present, then 0 will be returned.

In the following example, the POS function is used to select all rows where the literal string 'ABC' exists in the Name column:

```
SELECT *
FROM Country
WHERE POS('ABC' IN Name) > 0
```

POS or POSITION can only be used with string or memo columns or constants.

OCCURS Function

The OCCURS function returns the number of occurrences of one string within another string. The syntax is as follows:

```
OCCURS(string constant
       IN column_reference or string constant)
OCCURS(string constant,
       column_reference or string constant)
```

If the search string is not present, then 0 will be returned.

In the following example, the OCCURS function is used to select all rows where the literal string 'ABC' occurs at least once in the Name column:

```
SELECT *
FROM Country
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
WHERE OCCURS('ABC' IN Name) > 0
```

OCCURS can only be used with string or memo columns or constants.

REPLACE Function

The REPLACE function replaces all occurrences of a given string with a new string within another string. The syntax is as follows:

```
REPLACE(string constant WITH new string constant
        IN column_reference or string constant)
REPLACE(string constant,new string constant,
        column_reference or string constant)
```

If the search string is not present, then the result will be the original table column or string constant.

In the following example, the REPLACE function is used to replace all occurrences of 'Mexico' with 'South America':

```
UPDATE biolife
SET notes=REPLACE('Mexico' WITH 'South America' IN notes)
```

REPLACE can only be used with string or memo columns or constants.

REPEAT Function

The REPEAT function repeats a given string a specified number of times and returns the concatenated result. The syntax is as follows:

```
REPEAT(column_reference or string constant
       FOR number_of_occurrences)
REPEAT(column_reference or string constant,
       number_of_occurrences)
```

In the following example, the REPEAT function is used to replicate the dash (-) character 60 times to use as a separator in a multi-line string:

```
UPDATE biolife
SET notes='Notes'+#13+#10+
REPEAT('-' FOR 60)+#13+#10+#13+#10+
'These are the notes'
```

REPEAT can only be used with string or memo columns or constants.

CONCAT Function

The CONCAT function concatenates two strings together and returns the concatenated result. The syntax is as follows:

```
CONCAT(column_reference or string constant
       WITH column_reference or string constant)
CONCAT(column_reference or string constant,
       column_reference or string constant)
```

In the following example, the CONCAT function is used to concatenate two strings together:

```
UPDATE biolife
SET notes=CONCAT(Notes WITH #13+#10+#13+#10+'End of Notes')
```

CONCAT can only be used with string or memo columns or constants.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

5.6 Numeric Functions

Use numeric functions to manipulate numeric values in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following numeric functions:

Function	Description
ABS	Converts a number to its absolute value (non-negative).
ACOS	Returns the arccosine of a number as an angle expressed in radians.
ASIN	Returns the arcsine of a number as an angle expressed in radians.
ATAN	Returns the arctangent of a number as an angle expressed in radians.
ATAN2	Returns the arctangent of x and y coordinates as an angle expressed in radians.
CEIL or CEILING	Returns the lowest integer greater than or equal to a number.
COS	Returns the cosine of an angle.
COT	Returns the cotangent of an angle.
DEGREES	Converts a number representing radians into degrees.
EXP	Returns the exponential value of a number.
FLOOR	Returns the highest integer less than or equal to a number.
LOG	Returns the natural logarithm of a number.
LOG10	Returns the base 10 logarithm of a number.
MOD	Returns the modulus of two integers as an integer.
PI	Returns the ratio of a circle's circumference to its diameter - approximated as 3.1415926535897932385.
POWER	Returns the value of a base number raised to the specified power.
RADIANS	Converts a number representing degrees into radians.
RAND	Returns a random number.
ROUND	Rounds a number to a specified number of decimal places.
SIGN	Returns -1 if a number is less than 0, 0 if a number is 0, or 1 if a number is

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

	greater than 0.
SIN	Returns the sine of an angle.
SQRT	Returns the square root of a number.
TAN	Returns the tangent of an angle.
TRUNC or TRUNCATE	Truncates a numeric argument to the specified number of decimal places

ABS Function

The ABS function converts a numeric value to its absolute, or non-negative value:

```
ABS(column_reference or numeric constant)
```

ABS can only be used with numeric columns or constants.

ACOS Function

The ACOS function returns the arccosine of a number as an angle expressed in radians:

```
ACOS(column_reference or numeric constant)
```

ACOS can only be used with numeric columns or constants.

ASIN Function

The ASIN function returns the arcsine of a number as an angle expressed in radians:

```
ASIN(column_reference or numeric constant)
```

ASIN can only be used with numeric columns or constants.

ATAN Function

The ATAN function returns the arctangent of a number as an angle expressed in radians:

```
ATAN(column_reference or numeric constant)
```

ATAN can only be used with numeric columns or constants.

ATAN2 Function

The ATAN2 function returns the arctangent of x and y coordinates as an angle expressed in radians:

```
ATAN2(column_reference or numeric constant,
              column_reference or numeric constant)
```

ATAN2 can only be used with numeric columns or constants.

CEIL or CEILING Function

The CEIL or CEILING function returns the lowest integer greater than or equal to a number:

```
CEIL(column_reference or numeric constant)
CEILING(column_reference or numeric constant)
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

CEIL or CEILING can only be used with numeric columns or constants.

COS Function

The COS function returns the cosine of an angle:

```
COS(column_reference or numeric constant)
```

COS can only be used with numeric columns or constants.

COT Function

The COT function returns the cotangent of an angle:

```
COT(column_reference or numeric constant)
```

COT can only be used with numeric columns or constants.

DEGREES Function

The DEGREES function converts a number representing radians into degrees:

```
DEGREES(column_reference or numeric constant)
```

DEGREES can only be used with numeric columns or constants.

EXP Function

The EXP function returns the exponential value of a number:

```
EXP(column_reference or numeric constant)
```

EXP can only be used with numeric columns or constants.

FLOOR Function

The FLOOR function returns the highest integer less than or equal to a number:

```
FLOOR(column_reference or numeric constant)
```

FLOOR can only be used with numeric columns or constants.

LOG Function

The LOG function returns the natural logarithm of a number:

```
LOG(column_reference or numeric constant)
```

LOG can only be used with numeric columns or constants.

LOG10 Function

The LOG10 function returns the base 10 logarithm of a number:

```
LOG10(column_reference or numeric constant)
```

LOG10 can only be used with numeric columns or constants.

MOD Function

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The MOD function returns the modulus of two integers. The modulus is the remainder that is present when dividing the first integer by the second integer:

```
MOD(column_reference or integer constant,
            column_reference or integer constant)
```

MOD can only be used with integer columns or constants.

PI Function

The PI function returns the ratio of a circle's circumference to its diameter - approximated as 3.1415926535897932385:

```
PI()
```

POWER Function

The POWER function returns the value of a base number raised to the specified power:

```
POWER(column_reference or numeric constant
      TO column_reference or numeric constant)
POWER(column_reference or numeric constant,
      column_reference or numeric constant)
```

POWER can only be used with numeric columns or constants.

RADIANS Function

The RADIANS function converts a number representing degrees into radians:

```
RADIANS(column_reference or numeric constant)
```

RADIANS can only be used with numeric columns or constants.

RAND Function

The RAND function returns a random number:

```
RAND([RANGE range of random values])
```

The range value is optional and is used to limit the random numbers returned to between 0 and the range value specified. If the range is not specified then any number within the full range of numeric values may be returned.

ROUND Function

The ROUND function rounds a numeric value to a specified number of decimal places:

```
ROUND(column_reference or numeric constant
              [TO number of decimal places])
ROUND(column_reference or numeric constant
     [, number of decimal places])
```

The number of decimal places is optional, and if not specified the value returned will be rounded to 0 decimal places.

ROUND can only be used with numeric columns or constants.

Note

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The ROUND function performs "normal" rounding where the number is rounded up if the fractional portion beyond the number of decimal places being rounded to is greater than or equal to 5 and down if the fractional portion is less than 5. Also, if using the ROUND function with floating-point values, it is possible to encounter rounding errors due to the nature of floating-point values and their inability to accurately express certain numbers. If you want to eliminate this possibility you should use the CAST function to convert the floating-point column or constant to a BCD value (DECIMAL or NUMERIC data type in SQL). This will allow for the rounding to occur as desired since BCD values can accurately represent these numbers without errors.

SIGN Function

The SIGN function returns -1 if a number is less than 0, 0 if a number is 0, or 1 if a number is greater than 0:

```
SIGN(column_reference or numeric constant)
```

SIGN can only be used with numeric columns or constants.

SIN Function

The SIN function returns the sine of an angle:

```
SIN(column_reference or numeric constant)
```

SIN can only be used with numeric columns or constants.

SQRT Function

The SQRT function returns the square root of a number:

```
SQRT(column_reference or numeric constant)
```

SQRT can only be used with numeric columns or constants.

TAN Function

The TAN function returns the tangent of an angle:

```
TAN(column_reference or numeric constant)
```

TAN can only be used with numeric columns or constants.

TRUNC or TRUNCATE Function

The TRUNC or TRUNCATE function truncates a numeric value to a specified number of decimal places:

```
TRUNC(column_reference or numeric constant
      [TO number of decimal places])
TRUNCATE(column_reference or numeric constant
         [TO number of decimal places])
TRUNC(column_reference or numeric constant
      [, number of decimal places])
TRUNCATE(column_reference or numeric constant
         [, number of decimal places])
```

The number of decimal places is optional, and if not specified the value returned will be truncated to 0 decimal places.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

TRUNC or TRUNCATE can only be used with numeric columns or constants.

Note
If using the TRUNC or TRUNCATE function with floating-point values, it is possible to encounter truncation errors due to the nature of floating-point values and their inability to accurately express certain numbers. If you want to eliminate this possibility you should use the CAST function to convert the floating-point column or constant to a BCD value (DECIMAL or NUMERIC data type in SQL). This will allow for the truncation to occur as desired since BCD values can accurately represent these numbers without errors.

5.7 Boolean Functions

Use boolean functions to manipulate any values in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following boolean functions:

Function	Description
IF	Performs IF..ELSE type of inline expression handling.
IFNULL	Performs IF..ELSE type of inline expression handling specifically for NULL values.
NULLIF	Returns a NULL if two values are equivalent.
COALESCE	Returns the first non-NULL value from a list of expressions.

IF Function

The IF function performs inline IF..ELSE boolean expression handling:

```
IF(boolean expression THEN result expression
   ELSE result expression)
IF(boolean expression, result expression,
   result expression)
```

Both result expressions must be of the same data type. Use the CAST function to ensure that both expressions are of the same data type.

In the following example, if the Category column contains the value 'WRASSE', then the column value returned will be the Common_Name column, otherwise it will be the Species Name column:

```
SELECT IF(Upper(Category)='WRASSE'
THEN Common_Name
ELSE "Species Name") AS Name
FROM Biolife
```

The IF function can be used in WHERE clause comparisons to cause a conditional comparison:

```
SELECT *
FROM Employee
WHERE IF(LastName='Young' THEN PhoneExt='233' ELSE PhoneExt='22')
```

IFNULL Function

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The IFNULL function performs inline IF..ELSE boolean expression handling specifically on NULL values:

```
IFNULL(expression THEN result expression
       ELSE result expression)
IFNULL(expression, result expression,
       result expression)
```

Both result expressions must be of the same data type. Use the CAST function to ensure that both expressions are of the same data type.

In the following example, if the Category column contains a NULL value, then the column value returned will be the Common_Name column, otherwise it will be the Species Name column:

```
SELECT IFNULL(Category THEN Common_Name
ELSE "Species Name") AS Name
FROM Biolife
```

The IFNULL function can be used in WHERE clause comparisons to cause a conditional comparison:

```
SELECT *
FROM Employee
WHERE IFNULL(Salary THEN 10000 ELSE Salary) > 8000
```

NULLIF Function

The NULLIF function returns a NULL if the two values passed as parameters are equal:

```
NULLIF(expression,expression)
```

Both expressions must be of the same data type. Use the CAST function to ensure that both expressions are of the same data type.

In the following example, if the EmpNo column contains the value 14, then the value returned will be NULL, otherwise it will be the EmpNo column value:

```
SELECT NULLIF(EmpNo,14) AS EmpNo
FROM Orders
```

The NULLIF function can be used in WHERE clause comparisons to cause a conditional comparison:

```
SELECT *
FROM Employee
WHERE NULLIF(Salary,10000) > 8000
```

COALESCE Function

The COALESCE function returns the first non-NULL value from a list of expressions:

```
COALESCE(expression [, expression [, expression]])
```

All expressions must be of the same data type. Use the CAST function to ensure that all expressions are of the same data type.

In the following example, if the Category column contains a NULL value, then the column value returned will be the Common_Name column. If the Common_name column contains a NULL, then the literal string 'No Name' will be returned:

```
SELECT COALESCE(Category,Common_Name,'No Name') AS Name
FROM Biolife
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

5.8 Aggregate Functions

Use aggregate functions to perform aggregate calculations on values in SELECT queries containing a GROUP BY clause. DBISAM's SQL supports the following aggregate functions:

Function	Description
AVG	Averages all numeric values in a column.
COUNT	Counts the total number of rows or the number of rows where the specified column is not NULL.
MAX	Determines the maximum value in a column.
MIN	Determines the minimum value in a column.
STDDEV	Calculates the standard deviation of all numeric values in a column.
SUM	Totals all numeric values in a column.
RUNSUM	Totals all numeric values in a column in a running total.

AVG Function

The AVG function returns the average of the values in a specified column or expression. The syntax is as follows:

```
AVG(column_reference or expression)
```

Use AVG to calculate the average value for a numeric column. As an aggregate function, AVG performs its calculation aggregating values in the same column(s) across all rows in a dataset. The dataset may be the entire table, a filtered dataset, or a logical group produced by a GROUP BY clause. Column values of zero are included in the averaging, so values of 1, 2, 3, 0, 0, and 0 result in an average of 1. NULL column values are not counted in the calculation. The following is an example of using the AVG function to calculate the average order amount for all orders:

```
SELECT AVG(ItemsTotal)
FROM Orders
```

AVG returns the average of values in a column or the average of a calculation using a column performed for each row (a calculated field). The following example shows how to use the AVG function to calculate an average order amount and tax amount for all orders:

```
SELECT AVG(ItemsTotal) AS AverageTotal,
AVG(ItemsTotal * 0.0825) AS AverageTax
FROM Orders
```

When used with a GROUP BY clause, AVG calculates one value for each group. This value is the aggregation of the specified column for all rows in each group. The following example aggregates the average value for the ItemsTotal column in the Orders table, producing a subtotal for each company in the Customer table:

```
SELECT c."Company",
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
AVG(o."ItemsTotal") AS Average,
MAX(o."ItemsTotal") AS Biggest,
MIN(o."ItemsTotal") AS Smallest
FROM "Customer.dat" c, "Orders.dat" o
WHERE (c."CustNo" = o."CustNo")
GROUP BY c."Company"
ORDER BY c."Company"
```

AVG operates only on numeric values.

COUNT Function

The COUNT function returns the number of rows that satisfy a query's search condition or the number of rows where the specified column is not NULL. The syntax is as follows:

```
COUNT(column_reference or expression)
```

Use COUNT to count the number of rows retrieved by a SELECT statement. The SELECT statement may be a single-table or multi-table query. The value returned by COUNT reflects a reduced row count produced by a filtered dataset. The following example returns the total number of rows in the Averaging source table with a non-NULL Amount column:

```
SELECT COUNT(Amount)
FROM Averaging
```

The following example returns the total number of rows in the filtered Orders source table irrespective of any NULL column values:

```
SELECT COUNT(*)
FROM Orders
WHERE (Orders.ItemsTotal > 5000)
```

MAX Function

The MAX function returns the largest value in the specified column. The syntax is as follows:

```
MAX(column_reference or expression)
```

Use MAX to calculate the largest value for a string, numeric, date, time, or timestamp column. As an aggregate function, MAX performs its calculation aggregating values in the same column(s) across all rows in a dataset. The dataset may be the entire table, a filtered dataset, or a logical group produced by a GROUP BY clause. Column values of zero are included in the aggregation. NULL column values are not counted in the calculation. If the number of qualifying rows is zero, MAX returns a NULL value. The following is an example of using the MAX function to calculate the largest order amount for all orders:

```
SELECT MAX(ItemsTotal)
FROM Orders
```

MAX returns the largest value in a column or a calculation using a column performed for each row (a calculated field). The following example shows how to use the MAX function to calculate the largest order amount and tax amount for all orders:

```
SELECT MAX(ItemsTotal) AS HighestTotal,
MAX(ItemsTotal * 0.0825) AS HighestTax
FROM Orders
```

When used with a GROUP BY clause, MAX returns one calculation value for each group. This value is the aggregation of the specified column for all rows in each group. The following example aggregates the largest value for the ItemsTotal column in the Orders table, producing a subtotal for each company in

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

the Customer table:

```
SELECT c."Company",
AVG(o."ItemsTotal") AS Average,
MAX(o."ItemsTotal") AS Biggest,
MIN(o."ItemsTotal") AS Smallest
FROM "Customer.dat" c, "Orders.dat" o
WHERE (c."CustNo" = o."CustNo")
GROUP BY c."Company"
ORDER BY c."Company"
```

MAX can be used with all string, numeric, date, time, and timestamp columns. The return value is of the same type as the column.

MIN Function

The MIN function returns the smallest value in the specified column. The syntax is as follows:

```
MIN(column_reference or expression)
```

Use MIN to calculate the smallest value for a string, numeric, date, time, or timestamp column. As an aggregate function, MIN performs its calculation aggregating values in the same column(s) across all rows in a dataset. The dataset may be the entire table, a filtered dataset, or a logical group produced by a GROUP BY clause. Column values of zero are included in the aggregation. NULL column values are not counted in the calculation. If the number of qualifying rows is zero, MIN returns a NULL value. The following is an example of using the MIN function to calculate the smallest order amount for all orders:

```
SELECT MIN(ItemsTotal)
FROM Orders
```

MIN returns the smallest value in a column or a calculation using a column performed for each row (a calculated field). The following example shows how to use the MIN function to calculate the smallest order amount and tax amount for all orders:

```
SELECT MIN(ItemsTotal) AS LowestTotal,
MIN(ItemsTotal * 0.0825) AS LowestTax
FROM Orders
```

When used with a GROUP BY clause, MIN returns one calculation value for each group. This value is the aggregation of the specified column for all rows in each group. The following example aggregates the smallest value for the ItemsTotal column in the Orders table, producing a subtotal for each company in the Customer table:

```
SELECT c."Company",
AVG(o."ItemsTotal") AS Average,
MAX(o."ItemsTotal") AS Biggest,
MIN(o."ItemsTotal") AS Smallest
FROM "Customer.dat" c, "Orders.dat" o
WHERE (c."CustNo" = o."CustNo")
GROUP BY c."Company"
ORDER BY c."Company"
```

MIN can be used with all string, numeric, date, time, and timestamp columns. The return value is of the same type as the column.

STDDEV Function

The STDDEV function returns the standard deviation of the values in a specified column or expression. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
STDDEV(column_reference or expression)
```

Use STDDEV to calculate the standard deviation value for a numeric column. As an aggregate function, STDDEV performs its calculation aggregating values in the same column(s) across all rows in a dataset. The dataset may be the entire table, a filtered dataset, or a logical group produced by a GROUP BY clause. NULL column values are not counted in the calculation. The following is an example of using the STDDEV function to calculate the standard deviation for a set of test scores:

```
SELECT STDDEV(TestScore)
FROM Scores
```

When used with a GROUP BY clause, STDDEV calculates one value for each group. This value is the aggregation of the specified column for all rows in each group.

STDDEV operates only on numeric values.

SUM Function

The SUM function calculates the sum of values for a column. The syntax is as follows:

```
SUM(column_reference or expression)
```

Use SUM to sum all the values in the specified column. As an aggregate function, SUM performs its calculation aggregating values in the same column(s) across all rows in a dataset. The dataset may be the entire table, a filtered dataset, or a logical group produced by a GROUP BY clause. Column values of zero are included in the aggregation. NULL column values are not counted in the calculation. If the number of qualifying rows is zero, SUM returns a NULL value. The following is an example of using the SUM function to calculate the total order amount for all orders:

```
SELECT SUM(ItemsTotal)
FROM Orders
```

SUM returns the total sum of a column or a calculation using a column performed for each row (a calculated field). The following example shows how to use the SUM function to calculate the total order amount and tax amount for all orders:

```
SELECT SUM(ItemsTotal) AS Total,
SUM(ItemsTotal * 0.0825) AS TotalTax
FROM orders
```

When used with a GROUP BY clause, SUM returns one calculation value for each group. This value is the aggregation of the specified column for all rows in each group. The following example aggregates the total value for the ItemsTotal column in the Orders table, producing a subtotal for each company in the Customer table:

```
SELECT c."Company",
SUM(o."ItemsTotal") AS SubTotal
FROM "Customer.dat" c, "Orders.dat" o
WHERE (c."CustNo" = o."CustNo")
GROUP BY c."Company"
ORDER BY c."Company"
```

SUM operates only on numeric values.

RUNSUM Function

The RUNSUM function calculates the sum of values for a column in a running total. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
RUNSUM(column_reference or expression)
```

Use RUNSUM to sum all the values in the specified column in a continuous running total. The RUNSUM function is identical to the SUM function except for the fact that it does not reset itself when sub-totalling.

Note
The running total is only calculated according to the implicit order of the GROUP BY fields and is not affected by an ORDER BY statement.

5.9 AutoInc Functions

Use autoinc functions to return the last autoinc value from a given table in INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following autoinc functions:

Function	Description
LASTAUTOINC	Returns the last autoinc value from a specified table.
IDENT_CURRENT	Same as LASTAUTOINC, with a different name.

LASTAUTOINC Function

The LASTAUTOINC function returns the last autoinc value from a specified table. The syntax is as follows:

```
LASTAUTOINC(table name constant)
```

The LASTAUTOINC function will return the last autoinc value from the specified table relative to the start of the SQL statement currently referencing the LASTAUTOINC function. Because of this, it is possible for LASTAUTOINC to not return the most recent last autoinc value for the specified table. It is usually recommended that you only use this function within the scope of a transaction in order to guarantee that you have retrieved the correct last autoinc value from the table. The following example illustrates how this would be accomplished using an SQL script and a master-detail insert:

```
START TRANSACTION;

INSERT INTO customer (company) VALUES ('Test');

INSERT INTO orders (custno,empno) VALUES (LASTAUTOINC('customer'),100);
INSERT INTO orders (custno,empno) VALUES (LASTAUTOINC('customer'),200);

COMMIT FLUSH;
```

5.10 Text Search Functions

Use full text indexing functions to search for specific words in a given column in SELECT, INSERT,

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

UPDATE, or DELETE queries. The word search is controlled by the text indexing parameters for the table in which the column resides. DBISAM's SQL supports the following word search functions:

Function	Description
TEXTSEARCH	Performs an optimized text word search on a field, if the field is part of the full text index for the table, or a brute-force text word search if not.
TEXTOCCURS	Counts the number of times a list of words appears in a field based upon the full text indexing parameters for the table.

TEXTSEARCH Function

The TEXTSEARCH function searches a column for a given set of words in a search string constant. The syntax is as follows:

```
TEXTSEARCH(search string constant
           IN column_reference)
TEXTSEARCH(search string constant,
           column_reference)
```

The optimization of the TEXTSEARCH function is controlled by whether the column being searched is part of the full text index for the table in which the column resides. If the column is not part of the full text index then the search will resort to a brute-force scan of the contents of the column in every record that satisfies any prior conditions in the WHERE clause. Also, the parsing of the list of words in the search string constant is controlled by the text indexing parameters for the table in which the column being searched resides.

In the following example, the words 'DATABASE QUERY SPEED' are searched for in the TextBody column:

```
SELECT GroupNo, No
FROM article
WHERE TEXTSEARCH('DATABASE QUERY SPEED' IN TextBody)
```

TEXTSEARCH returns a boolean value indicating whether the list of words exists in the column for a given record. TEXTSEARCH can only be used with string or memo columns.

TEXTOCCURS Function

The TEXTOCCURS function searches a column for a given set of words in a search string constant and returns the number of times the words occur in the column. The syntax is as follows:

```
TEXTOCCURS(search string constant
           IN column_reference)
TEXTOCCURS(search string constant,
           column_reference)
```

TEXTOCCURS is always a brute-force operation and accesses the actual column contents to perform its functionality, unlike the TEXTSEARCH function which can be optimized by adding the column being searched to the full text index for the table. Also, the parsing of the list of words in the search string constant is controlled by the text indexing parameters for the table in which the column being searched resides.

In the following example, the number of occurrences of the words 'DATABASE QUERY SPEED' in the TextBody column are used to order the results of a TEXTSEARCH query in order to provide ranking for

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

the text search:

```
SELECT GroupNo, No,
TEXTOCCURS('DATABASE QUERY SPEED' IN TextBody) AS NumOccurs
FROM article
WHERE TEXTSEARCH('DATABASE QUERY SPEED' IN TextBody)
ORDER BY 3 DESC
```

TEXTOCCURS returns an integer value indicating the total number of times the list of words occurs in the column for a given record. TEXTOCCURS can only be used with string or memo columns.

5.11 Data Conversion Functions

Use data conversion functions to convert values from one type to another in SELECT, INSERT, UPDATE, or DELETE queries. DBISAM's SQL supports the following data conversion functions:

Function	Description
EXTRACT	Extracts the year, month, week, day of week, or day value of a date or the hours, minutes, or seconds value of a time.
CAST	Converts a given data value from one data type to another.
YEARSFROMMSECS	Takes milliseconds and returns the number of years.
DAYSFROMMSECS	Takes milliseconds and returns the number of days (as a remainder of the above years, not as an absolute).
HOURSFROMMSECS	Takes milliseconds and returns the number of hours (as a remainder of the above years and days, not as an absolute).
MINSFROMMSECS	Takes milliseconds and returns the number of minutes (as a remainder of the above years, days, and hours, not as an absolute).
SECSFROMMSECS	Takes milliseconds and returns the number of seconds (as a remainder of the above years, days, hours, and minutes, not as an absolute).
MSECSFROMMSECS	Takes milliseconds and returns the number of milliseconds (as a remainder of the above years, days, hours, minutes, and seconds, not as an absolute).

EXTRACT Function

The EXTRACT function returns a specific value from a date, time, or timestamp value. The syntax is as follows:

```
EXTRACT(extract_value
        FROM column_reference or expression)
EXTRACT(extract_value,
        column_reference or expression)
```

Use EXTRACT to return the year, month, week, day of week, day, hours, minutes, seconds, or

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

milliseconds from a date, time, or timestamp column. EXTRACT returns the value for the specified element as an integer.

The extract_value parameter may contain any one of the specifiers:

```
YEAR
MONTH
WEEK
DAYOFWEEK
DAYOFYEAR
DAY
HOUR
MINUTE
SECOND
MSECOND
```

The specifiers YEAR, MONTH, WEEK, DAYOFWEEK, DAYOFYEAR, and DAY can only be used with date and timestamp columns. The following example shows how to use the EXTRACT function to display the various elements of the SaleDate column:

```
SELECT SaleDate,
EXTRACT(YEAR FROM SaleDate) AS YearNo,
EXTRACT(MONTH FROM SaleDate) AS MonthNo,
EXTRACT(WEEK FROM SaleDate) AS WeekNo,
EXTRACT(DAYOFWEEK FROM SaleDate) AS WeekDayNo,
EXTRACT(DAYOFYEAR FROM SaleDate) AS YearDayNo,
EXTRACT(DAY FROM SaleDate) AS DayNo
FROM Orders
```

The following example uses a DOB column (containing birthdates) to filter those rows where the date is in the month of May. The month field from the DOB column is retrieved using the EXTRACT function and compared to 5, May being the fifth month:

```
SELECT DOB, LastName, FirstName
FROM People
WHERE (EXTRACT(MONTH FROM DOB) = 5)
```

Note
The WEEK and DAYOFWEEK parameters will return the week number and the day of the week according to ANSI/ISO standards. This means that the first week of the year (week 1) is the first week that contains the first Thursday in January and January 4th and the first day of the week (day 1) is Monday. Also, while ANSI-standard SQL provides the EXTRACT function specifiers TIMEZONE_HOUR and TIMEZONE_MINUTE, these specifiers are not supported in DBISAM's SQL.

EXTRACT operates only on date, time, and timestamp values.

CAST Function

The CAST function converts a specified value to the specified data type. The syntax is as follows:

```
CAST(column_reference AS data_type)
CAST(column_reference,data_type)
```

Use CAST to convert the value in the specified column to the data type specified. CAST can also be applied to literal and calculated values. CAST can be used in the columns list of a SELECT statement, in the predicate for a WHERE clause, or to modify the update atom of an UPDATE statement.

The data type parameter may be any valid SQL data type that is valid as a destination type for the source data being converted.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The statement below converts a timestamp column value to a date column value:

```
SELECT CAST(SaleDate AS DATE)
FROM ORDERS
```

Converting a column value with CAST allows use of other functions or predicates on an otherwise incompatible data type, such as using the SUBSTRING function on a date column:

```
SELECT SaleDate,
SUBSTRING(CAST(CAST(SaleDate AS DATE) AS CHAR(10)) FROM 1 FOR 1)
FROM Orders
```

Note
All conversions of dates or timestamps to strings are done using the 24-hour clock (military time).

YEARSFROMMSECS Function

The YEARSFROMMSECS function takes milliseconds and returns the number of years. The syntax is as follows:

```
YEARSFROMMSECS(column_reference or expression)
```

Use YEARSFROMMSECS to return the number of years contained in a milliseconds value as an integer value.

DAYSFROMMSECS Function

The DAYSFROMMSECS function takes milliseconds and returns the number of days as a remainder of the number of years present in the milliseconds. The syntax is as follows:

```
DAYSFROMMSECS(column_reference or expression)
```

Use DAYSFROMMSECS to return the number of days present in a milliseconds value as an integer value. The number of days is represented as the remainder of days once the number of years is removed from the milliseconds value using the YEARSFROMMSECS function.

HOURSFROMMSECS Function

The HOURSFROMMSECS function takes milliseconds and returns the number of hours as a remainder of the number of years and days present in the milliseconds. The syntax is as follows:

```
HOURSFROMMSECS(column_reference or expression)
```

Use HOURSFROMMSECS to return the number of hours present in a milliseconds value as an integer value. The number of hours is represented as the remainder of hours once the number of years and days is removed from the milliseconds value using the YEARSFROMMSECS and DAYSFROMMSECS functions.

MINSFROMMSECS Function

The MINSFROMMSECS function takes milliseconds and returns the number of minutes as a remainder of the number of years, days, and hours present in the milliseconds. The syntax is as follows:

```
MINSFROMMSECS(column_reference or expression)
```

Use MINSFROMMSECS to return the number of minutes present in a milliseconds value as an integer value. The number of minutes is represented as the remainder of minutes once the number of years, days, and hours is removed from the milliseconds value using the YEARSFROMMSECS,

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

DAYSFROMMSECS, and HOURSFROMMSECS functions.

SECSFROMMSECS Function

The SECSFROMMSECS function takes milliseconds and returns the number of seconds as a remainder of the number of years, days, hours, and minutes present in the milliseconds. The syntax is as follows:

```
SECSFROMMSECS(column_reference or expression)
```

Use SECSFROMMSECS to return the number of seconds present in a milliseconds value as an integer value. The number of seconds is represented as the remainder of seconds once the number of years, days, hours, and minutes is removed from the milliseconds value using the YEARSFROMMSECS, DAYSFROMMSECS, HOURSFROMMSECS, and MINSFROMMSECS functions.

MSECSFROMMSECS Function

The MSECSFROMMSECS function takes milliseconds and returns the number of milliseconds as a remainder of the number of years, days, hours, minutes, and seconds present in the milliseconds. The syntax is as follows:

```
MSECSFROMMSECS(column_reference or expression)
```

Use MSECSFROMMSECS to return the number of milliseconds present in a milliseconds value as an integer value. The number of milliseconds is represented as the remainder of milliseconds once the number of years, days, hours, minutes, and seconds is removed from the milliseconds value using the YEARSFROMMSECS, DAYSFROMMSECS, HOURSFROMMSECS, MINSFROMMSECS, and SECSFROMMSECS functions.

5.12 Constant Functions

Use the following constant functions to obtain the literal values that each of them returns:

Function	Description
CURRENT_DATE	Returns the current system date.
CURRENT_TIME	Returns the current system time.
CURRENT_TIMESTAMP	Returns the current system date and time.
CURRENT_GUID	Returns a unique 38-byte GUID value in string format.
CURRENT_USER	Returns the User ID that is being used by Alventis to access the Server that this query belongs to. This function is an Alventis extension and may not be supported by other systems.

All of these functions take no parameters, so you do not need to write empty parenthesis after them, for example:

```
SELECT CURRENT_DATE AS TodaysDate, ID FROM TableName
```

The date/time functions will return the Server's date/time if executed by a Remote Server (i.e., when the query belongs to a Remote Database).

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

5.13 SELECT Statement

Introduction

The SQL SELECT statement is used to retrieve data from tables. You can use the SELECT statement to:

- Retrieve a single row, or part of a row, from a table, referred to as a singleton select.
- Retrieve multiple rows, or parts of rows, from a table.
- Retrieve related rows, or parts of rows, from a join of two or more tables.

Syntax

```
SELECT [DISTINCT | ALL] * | column
[AS correlation_name | correlation_name], [column...]

[INTO destination_table]

FROM table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE]

[[[[INNER | [LEFT | RIGHT] OUTER JOIN] table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE]
ON join_condition]

[WHERE predicates]

[GROUP BY group_list]

[HAVING predicates]

[[UNION | EXCEPT| INTERSECT] [ALL] [SELECT...]]

[ORDER BY order_list [NOCASE]]

[TOP number_of_rows]

[LOCALE locale_name | LOCALE CODE locale_code]

[ENCRYPTED WITH password]

[NOJOINOPTIMIZE]
[JOINOPTIMIZECOSTS]
```

The SELECT clause defines the list of items returned by the SELECT statement. The SELECT clause uses a comma-separated list composed of: table columns, literal values, and column or literal values modified by functions. You cannot use parameters in this list of items. Use an asterisk to retrieve values from all columns. Columns in the column list for the SELECT clause may come from more than one table, but can only come from those tables listed in the FROM clause. The FROM clause identifies the table(s) from which data is retrieved.

The following example retrieves data for two columns in all rows of a table:

```
SELECT CustNo, Company
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
FROM Orders
```

You can use the AS keyword to specify a column correlation name, or alternately you can simply just specify the column correlation name after the selected column. The following example uses both methods to give each selected column a more descriptive name in the query result set:

```
SELECT Customer.CustNo AS "Customer #",
Customer.Company AS "Company Name",
Orders.OrderNo "Order #",
SUM(Items.Qty) "Total Qty"
FROM Customer LEFT OUTER JOIN Orders ON Customer.Custno=Orders.Custno
LEFT OUTER JOIN Items ON Orders.OrderNo=Items.OrderNo
WHERE Customer.Company LIKE '%Diver%'
GROUP BY 1,2
ORDER BY 1
```

Use DISTINCT to limit the retrieved data to only distinct rows. The distinctness of rows is based on the combination of all of the columns in the SELECT clause columns list. DISTINCT can only be used with simple column types like string and integer; it cannot be used with complex column types like blob.

INTO Clause

The INTO clause specifies a table into which the query results are generated. The syntax is as follows:

```
INTO destination_table
```

Use an INTO clause to specify the table where the query results will be stored when the query has completed execution. The following example shows how to generate all of the orders in the month of January as a table on disk named "Results":

```
SELECT *
INTO "Results"
FROM "Orders"
```

If you do not specify a drive and directory in the destination table name, for local sessions, or a database name in the destination table name, for remote sessions, then the destination table will be created in the current active database for the query being executed.

The following examples show the different options for the INTO clause and their resultant destination table names.

This example produces a destination table in the current database called "Results":

```
SELECT *
INTO "Results"
FROM "Orders"
```

This example produces a destination table called "Results" in the specified local database directory (valid for local sessions only):

```
SELECT *
INTO "c:\MyData\Results"
FROM "Orders"
```

This example produces a destination table called "Results" in the specified database (valid for remote sessions only):

```
SELECT *
INTO "\MyRemoteDB\Results"
FROM "Orders"
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

This example produces an in-memory destination table called "Results":

```
SELECT *
INTO "\Memory\Results"
FROM "Orders"
```

Note
There are some important caveats when using the INTO clause:

- The INTO clause creates the resultant table from scratch, so if a table with the same name in the same location already exists, it will be overwritten. This also means that any indexes defined for the table will be removed or modified, even if the result set columns match those of the existing table.

- You must make sure that you close the query before trying to access the destination table with another table component. If you do not an exception will be raised.

- You must make sure to delete the table after you are done if you don't wish to leave it on disk or in-memory for further use.

- Remote sessions can only produce tables that are accessible from the database server and cannot automatically create a local table from a query on the database server by specifying a local path for the INTO clause. The path for the INTO clause must be accessible from the database server in order for the query to be successfully executed.

- The destination table cannot be passed to the INTO clause via a parameter.

FROM Clause

The FROM clause specifies the tables from which a SELECT statement retrieves data. The syntax is as follows:

```
FROM table_reference [AS] [correlation_name]
[, table_reference...]
```

Use a FROM clause to specify the table or tables from which a SELECT statement retrieves data. The value for a FROM clause is a comma-separated list of table names. Specified table names must follow DBISAM's SQL naming conventions for tables. Please see the Naming Conventions topic for more information. The following SELECT statement below retrieves data from a single table:

```
SELECT *
FROM "Customer"
```

The following SELECT statement below retrieves data from a single in-memory table:

```
SELECT *
FROM "\Memory\Customer"
```

You can use the AS keyword to specify a table correlation name, or alternately you can simply just specify the table correlation name after the source table name. The following example uses both methods to give each source table a shorter name to be used in qualifying source columns in the query:

```
SELECT c.CustNo AS "Customer #",
c.Company AS "Company Name",
o.OrderNo "Order #",
SUM(i.Qty) "Total Qty"
FROM Customer AS c LEFT OUTER JOIN Orders AS o ON c.Custno=o.Custno
LEFT OUTER JOIN Items i ON o.OrderNo=i.OrderNo
WHERE c.Company LIKE '%Diver%'
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
GROUP BY 1,2
ORDER BY 1
```

Use the EXCLUSIVE keyword to specify that the table should be opened exclusively.

Note
Be careful when using the EXCLUSIVE keyword with a table that is specified more than once in the same query, as is the case with recursive relationships between a table and itself.

See the section below entitled JOIN clauses for more information on retrieving data from multiple tables in a single SELECT query.

The table reference cannot be passed to a FROM clause via a parameter.

JOIN Clauses

There are three types of JOIN clauses that can be used in the FROM clause to perform relational joins between source tables. The implicit join condition is always Cartesian for source tables without an explicit JOIN clause.

Join Type	Description
Cartesian	Joins two tables, matching each row of one table with each row from the other.
INNER	Joins two tables, filtering out non-matching rows.
OUTER	Joins two tables, retaining non-matching rows.

Cartesian Join

A Cartesian join connects two tables in a non-relational manner. The syntax is as follows:

```
FROM table_reference, table_reference [,table_reference...]
```

Use a Cartesian join to connect the column of two tables into one result set, but without correlation between the rows from the tables. Cartesian joins match each row of the source table with each row of the joining table. No column comparisons are used, just simple association. If the source table has 10 rows and the joining table has 10, the result set will contain 100 rows as each row from the source table is joined with each row from the joined table.

INNER JOIN Clause

An INNER join connects two tables based on column values common between the two, excluding non-matches. The syntax is as follows:

```
FROM table_reference
[INNER] JOIN table_reference ON predicate
[[INNER] JOIN table_reference ON predicate...]
```

Use an INNER JOIN to connect two tables, a source and joining table, that have values from one or more columns in common. One or more columns from each table are compared in the ON clause for equal values. For rows in the source table that have a match in the joining table, the data for the source table rows and matching joining table rows are included in the result set. Rows in the source table without matches in the joining table are excluded from the joined result set. In the following example the Customer and Orders tables are joined based on values in the CustNo column, which each table contains:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
SELECT *
FROM Customer c INNER JOIN Orders o ON (c.CustNo=o.CustNo)
```

More than one table may be joined with an INNER JOIN. One use of the INNER JOIN operator and corresponding ON clause is required for each set of two tables joined. One columns comparison predicate in an ON clause is required for each column compared to join each two tables. The following example joins the Customer table to Orders, and then Orders to Items. In this case, the joining table Orders acts as a source table for the joining table Items:

```
SELECT *
FROM Customer c JOIN Orders o ON (c.CustNo = o.CustNo)
JOIN Items i ON (o.OrderNo = i.OrderNo)
```

Tables may also be joined using a concatenation of multiple column values to produce a single value for the join comparison predicate. In the following example the ID1 and ID2 columns in the Joining table are concatenated and compared with the values in the single column ID in Source:

```
SELECT *
FROM Source s INNER JOIN Joining j ON (s.ID = j.ID1 || j.ID2)
```

OUTER JOIN Clause

The OUTER JOIN clause connects two tables based on column values common between the two, including non-matches. The syntax is as follows:

```
FROM table_reference LEFT | RIGHT [OUTER]
JOIN table_reference ON predicate
[LEFT | RIGHT [OUTER] JOIN table_reference ON predicate...]
```

Use an OUTER JOIN to connect two tables, a source and joining table, that have one or more columns in common. One or more columns from each table are compared in the ON clause for equal values. The primary difference between inner and outer joins is that, in outer joins rows from the source table that do not have a match in the joining table are not excluded from the result set. Columns from the joining table for rows in the source table without matches have NULL values.

In the following example the Customer and Orders tables are joined based on values in the CustNo column, which each table contains. For rows from Customer that do not have a matching value between Customer.CustNo and Orders.CustNo, the columns from Orders contain NULL values:

```
SELECT *
FROM Customer c LEFT OUTER JOIN Orders o ON (c.CustNo = o.CustNo)
```

The LEFT modifier causes all rows from the table on the left of the OUTER JOIN operator to be included in the result set, with or without matches in the table to the right. If there is no matching row from the table on the right, its columns contain NULL values. The RIGHT modifier causes all rows from the table on the right of the OUTER JOIN operator to be included in the result set, with or without matches. If there is no matching row from the table on the left, its columns contain NULL values.

More than one table may be joined with an OUTER JOIN. One use of the OUTER JOIN operator and corresponding ON clause is required for each set of two tables joined. One column comparison predicate in an ON clause is required for each column compared to join each two tables. The following example joins the Customer table to the Orders table, and then Orders to Items. In this case, the joining table Orders acts as a source table for the joining table Items:

```
SELECT *
FROM Customer c LEFT OUTER JOIN Orders o ON (c.CustNo = o.CustNo)
LEFT OUTER JOIN Items i ON (o.OrderNo = i.OrderNo)
```

Tables may also be joined using expressions to produce a single value for the join comparison predicate.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

In the following example the ID1 and ID2 columns in Joining are separately compared with two values produced by the SUBSTRING function using the single column ID in Source:

```
SELECT *
FROM Source s RIGHT OUTER JOIN Joining j
ON (SUBSTRING(s.ID FROM 1 FOR 2) = j.ID1) AND
(SUBSTRING(s.ID FROM 3 FOR 1) = j.ID2)
```

WHERE Clause

The WHERE clause specifies filtering conditions for the SELECT statement. The syntax is as follows:

```
WHERE predicates
```

Use a WHERE clause to limit the effect of a SELECT statement to a subset of rows in the table, and the clause is optional.

The value for a WHERE clause is one or more logical expressions, or predicates, that evaluate to true or false for each row in the table. Only those rows where the predicates evaluate to TRUE are retrieved by the SELECT statement. For example, the SELECT statement below retrieves all rows where the State column contains a value of 'CA':

```
SELECT Company, State
FROM Customer
WHERE State='CA'
```

A column used in the WHERE clause of a statement is not required to also appear in the SELECT clause of that statement. In the preceding statement, the State column could be used in the WHERE clause even if it was not also in the SELECT clause.

Multiple predicates must be separated by one of the logical operators OR or AND. Each predicate can be negated with the NOT operator. Parentheses can be used to isolate logical comparisons and groups of comparisons to produce different row evaluation criteria. For example, the SELECT statement below retrieves all rows where the State column contains a value of 'CA' or a value of 'HI':

```
SELECT Company, State
FROM Customer
WHERE (State='CA') OR (State='HI')
```

Subqueries are supported in the WHERE clause. A subquery works like a search condition to restrict the number of rows returned by the outer, or "parent" query. Such subqueries must be valid SELECT statements. SELECT subqueries cannot be correlated in DBISAM's SQL, i.e. they cannot refer to columns in the outer (or "parent") statement. In the following statement, the subquery is said to be un-correlated:

```
SELECT *
FROM "Clients" C
WHERE C.Acct_Nbr IN
  (SELECT H.Acct_Nbr
  FROM "Holdings" H
  WHERE H.Pur_Date BETWEEN '1994-01-01' AND '1994-12-31')
```

Note
Column correlation names cannot be used in filter comparisons in the WHERE clause. Use the actual column name instead.

A WHERE clause filters data prior to the aggregation of a GROUP BY clause. For filtering based on aggregated values, use a HAVING clause.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Columns devoid of data contain NULL values. To filter using such column values, use the IS NULL predicate.

GROUP BY Clause

The GROUP BY clause combines rows with column values in common into single rows for the SELECT statement. The syntax is as follows:

```
GROUP BY column_reference [, column reference...]
```

Use a GROUP BY clause to cause an aggregation process to be repeated once for each group of similar rows. Similarity between rows is determined by the distinct values (or combination of values) in the columns specified in the GROUP BY. For instance, a query with a SUM function produces a result set with a single row with the total of all the values for the column used in the SUM function. But when a GROUP BY clause is added, the SUM function performs its summing action once for each group of rows. In statements that support a GROUP BY clause, the use of a GROUP BY clause is optional. A GROUP BY clause becomes necessary when both aggregated and non-aggregated columns are included in the same SELECT statement.

In the statement below, the SUM function produces one subtotal of the ItemsTotal column for each distinct value in the CustNo column (i.e., one subtotal for each different customer):

```
SELECT CustNo, SUM(ItemsTotal)
FROM Orders
GROUP BY CustNo
```

The value for the GROUP BY clause is a comma-separated list of columns. Each column in this list must meet the following criteria:

- Be in one of the tables specified in the FROM clause of the query.
- Also be in the SELECT clause of the query.
- Cannot have an aggregate function applied to it (in the SELECT clause).
- Cannot be a BLOB column.

When a GROUP BY clause is used, all table columns in the SELECT clause of the query must meet at least one of the following criteria, or it cannot be included in the SELECT clause:

- Be in the GROUP BY clause of the query.
- Be the subject of an aggregate function.

Literal values in the SELECT clause are not subject to the preceding criteria and are not required to be in the GROUP BY clause in addition to the SELECT clause.

The distinctness of rows is based on the columns in the column list specified. All rows with the same values in these columns are combined into a single row (or logical group). Columns that are the subject of an aggregate function have their values across all rows in the group combined. All columns not the subject of an aggregate function retain their value and serve to distinctly identify the group. For example, in the SELECT statement below, the values in the Sales column are aggregated (totalled) into groups based on distinct values in the Company column. This produces total sales for each company:

```
SELECT C.Company, SUM(O.ItemsTotal) AS TotalSales
FROM Customer C, Orders O
WHERE C.CustNo=O.CustNo
GROUP BY C.Company
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
ORDER BY C.Company
```

A column may be referenced in a GROUP BY clause by a column correlation name, instead of actual column names. The statement below forms groups using the first column, Company, represented by the column correlation name Co:

```
SELECT C.Company Co, SUM(O.ItemsTotal) AS TotalSales
FROM Customer C, Orders O
WHERE C.CustNo=O.CustNo
GROUP BY Co
ORDER BY 1
```

HAVING Clause

The HAVING clause specifies filtering conditions for a SELECT statement. The syntax is as follows:

```
HAVING predicates
```

Use a HAVING clause to limit the rows retrieved by a SELECT statement to a subset of rows where aggregated column values meet the specified criteria. A HAVING clause can only be used in a SELECT statement when:

- The statement also has a GROUP BY clause.
- One or more columns are the subjects of aggregate functions.

The value for a HAVING clause is one or more logical expressions, or predicates, that evaluate to true or false for each aggregate row retrieved from the table. Only those rows where the predicates evaluate to true are retrieved by a SELECT statement. For example, the SELECT statement below retrieves all rows where the total sales for individual companies exceed $1,000:

```
SELECT Company, SUM(sales) AS TotalSales
FROM Sales1998
GROUP BY Company
HAVING (SUM(sales) >= 1000)
ORDER BY Company
```

Multiple predicates must be separated by one of the logical operators OR or AND. Each predicate can be negated with the NOT operator. Parentheses can be used to isolate logical comparisons and groups of comparisons to produce different row evaluation criteria.

A SELECT statement can include both a WHERE clause and a HAVING clause. The WHERE clause filters the data to be aggregated, using columns not the subject of aggregate functions. The HAVING clause then further filters the data after the aggregation, using columns that are the subject of aggregate functions. The SELECT query below performs the same operation as that above, but data is limited to those rows where the State column is 'CA':

```
SELECT Company, SUM(sales) AS TotalSales
FROM Sales1998
WHERE (State = 'CA')
GROUP BY Company
HAVING (TOTALSALES >= 1000)
ORDER BY Company
```

A HAVING clause filters data after the aggregation of a GROUP BY clause. For filtering based on row values prior to aggregation, use a WHERE clause.

UNION, EXCEPT, or INTERSECT Clause

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The UNION clause concatenates the rows of one query result set to the end of another query result set and returns the resultant rows. The EXCEPT clause returns all of the rows from one query result set that are not present in another query result set. The INTERSECT clause returns all of the rows from one query result set that are also present in another query result set. The syntax is as follows:

```
[[UNION | EXCEPT| INTERSECT] [ALL] [SELECT...]]
```

The SELECT statement for the source and destination query result sets must include the same number of columns for them to be UNION/EXCEPT/INTERSECT-compatible. The source table structures themselves need not be the same as long as those columns included in the SELECT statements are:

```
SELECT CustNo, Company
FROM Customers
EXCEPT
SELECT OldCustNo, OldCompany
FROM Old_Customers
```

The data types for all columns retrieved by the UNION/EXCEPT/INTERSECT across the multiple query result sets must be identical. If there is a data type difference between two query result sets for a given column, an error will occur. The following query shows how to handle such a case to avoid an error:

```
SELECT S.ID, CAST(S.Date_Field AS TIMESTAMP)
FROM Source S
UNION ALL
SELECT J.ID, J.Timestamp_Field
FROM Joiner J
```

Matching names is not mandatory for result set columns retrieved by the UNION/EXCEPT/INTERSECT across the multiple query result sets. Column name differences between the multiple query result sets are automatically handled. If a column in two query result sets has a different name, the column in the UNION/EXCEPT/INTERSECTed result set will use the column name from the first SELECT statement.

By default, non-distinct rows are aggregated into single rows in a UNION/EXCEPT/INTERSECT join. Use ALL to retain non-distinct rows.

Note
When using the EXCEPT or INTERSECT clauses with the ALL keyword, the resultant rows will reflect the total counts of duplicate matching rows in both query result sets. For example, if using EXCEPT ALL with a query result set that has two 'A' rows and a query result set that has 1 'A' row, the result set will contain 1 'A' row (1 matching out of the 2). The same is true with INTERSECT. If using INTERSECT ALL with a query result set that has three 'A' rows and a query result set that has 2 'A' rows, the result set will contain 2 'A' rows (2 matching out of the 3).

To join two query result sets with UNION/EXCEPT/INTERSECT where one query does not have a column included by another, a compatible literal or expression may be used instead in the SELECT statement missing the column. For example, if there is no column in the Joining table corresponding to the Name column in Source an expression is used to provide a value for a pseudo Joining.Name column. Assuming Source.Name is of type CHAR(10), the CAST function is used to convert an empty character string to CHAR(10):

```
SELECT S.ID, S.Name
FROM Source S
INTERSECT
SELECT J.ID, CAST('' AS CHAR(10))
FROM Joiner J
```

If using an ORDER BY or TOP clause, these clauses must be specified after the last SELECT statement

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

being joined with a UNION/EXCEPT/INTERSECT clause. The WHERE, GROUP BY, HAVING, LOCALE, ENCRYPTED, and NOJOINOPTIMIZE clauses can be specified for all or some of the individual SELECT statements being joined with a UNION/EXCEPT/INTERSECT clause. The INTO clause can only be specified for the first SELECT statement in the list of unioned SELECT statements. The following example shows how you could join two SELECT statements with a UNION clause and order the final joined result set:

```
SELECT CustNo, Company
FROM Customers
UNION
SELECT OldCustNo, Company
FROM Old_Customers
ORDER BY CustNo
```

When referring to actual column names in the ORDER BY clause you must use the column name of the first SELECT statement being joined with the UNION/EXCEPT/INTERSECT clause.

ORDER BY Clause

The ORDER BY clause sorts the rows retrieved by a SELECT statement. The syntax is as follows:

```
ORDER BY column_reference [ASC|DESC]
[, column_reference...[ASC|DESC]] [NOCASE]
```

Use an ORDER BY clause to sort the rows retrieved by a SELECT statement based on the values from one or more columns. In SELECT statements, use of this clause is optional.

The value for the ORDER BY clause is a comma-separated list of column names. The columns in this list must also be in the SELECT clause of the query statement. Columns in the ORDER BY list can be from one or multiple tables. If the columns used for an ORDER BY clause come from multiple tables, the tables must all be those that are part of a join. They cannot be a table included in the statement only through a SELECT subquery.

BLOB columns cannot be used in the ORDER BY clause.

A column may be specified in an ORDER BY clause using a number representing the relative position of the column in the SELECT of the statement. Column correlation names can also be used in an ORDER BY clause columns list. Calculations cannot be used directly in an ORDER BY clause. Instead, assign a column correlation name to the calculation and use that name in the ORDER BY clause.

Use ASC (or ASCENDING) to force the sort to be in ascending order (smallest to largest), or DESC (or DESCENDING) for a descending sort order (largest to smallest). When not specified, ASC is the implied default.

Use NOCASE to force the sort to be case-insensitive. This is also useful for allowing a live result set when an index is available that matches the ORDER BY clause but is marked as case-insensitive. When not specified, case-sensitive is the implied default.

The statement below sorts the result set ascending by the year extracted from the LastInvoiceDate column, then descending by the State column, and then ascending by the uppercase conversion of the Company column:

```
SELECT EXTRACT(YEAR FROM LastInvoiceDate) AS YY,
State,
UPPER(Company)
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
FROM Customer
ORDER BY YY DESC, State ASC, 3
```

TOP Clause

The TOP clause causes the query to only return the top N number of rows, respecting any GROUP BY, HAVING, or ORDER BY clauses. The syntax is as follows:

```
TOP number_of_rows
```

Use a TOP clause to only extract a certain number of rows in a SELECT statement, based upon any GROUP BY, HAVING, or ORDER BY clauses. The rows that are selected start at the logical top of the result set and proceed to the total number of rows matching the TOP clause. In SELECT statements, use of the clause is optional.

LOCALE Clause

Use a LOCALE clause to set the locale of a result set created by a canned query (not live). The syntax is:

```
LOCALE locale_name | LOCALE CODE locale_code
```

If this clause is not used, the default locale of any canned result set is based upon the locale of the first table in the FROM clause of the SELECT statement.

ENCRYPTED WITH Clause

The ENCRYPTED WITH clause causes a SELECT statement returning a canned result set to encrypt the temporary table on disk used for the result set with the specified password. The syntax is as follows:

```
ENCRYPTED WITH password
```

Use an ENCRYPTED WITH clause to force the temporary table created by a SELECT statement that returns a canned result set to be encrypted with the specified password. This clause can also be used to encrypt the contents of a table created by a SELECT statement that uses the INTO clause.

NOJOINOPTIMIZE Clause

The NOJOINOPTIMIZE clause causes all join re-ordering to be turned off for a SELECT statement. The syntax is as follows:

```
NOJOINOPTIMIZE
```

Use a NOJOINOPTIMIZE clause to force the query optimizer to stop re-ordering joins for a SELECT statement. In certain rare cases the query optimizer might not have enough information to know that re-ordering the joins will result in worse performance than if the joins were left in their original order, so in such cases you can include this clause to force the query optimizer to not perform the join re-ordering.

JOINOPTIMIZECOSTS Clause

The JOINOPTIMIZECOSTS clause causes the optimizer to take into account I/O costs when optimizing join expressions. The syntax is as follows:

```
JOINOPTIMIZECOSTS
```

Use a JOINOPTIMIZECOSTS clause to force the query optimizer to use I/O cost projections to determine

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

the most efficient way to process the conditions in a join expression. If you have a join expression with multiple conditions in it, then using this clause may help improve the performance of the join expression, especially if it is already executing very slowly.

5.14 INSERT Statement

Introduction

The SQL INSERT statement is used to add one or more new rows of data in a table.

Syntax

```
INSERT INTO table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE]

[(columns_list)]

VALUES (update_values) | SELECT statement

[COMMIT [INTERVAL commit_interval] [FLUSH]]
```

Use the INSERT statement to add new rows of data to a single table. Use a table reference in the INTO clause to specify the table to receive the incoming data. Use the EXCLUSIVE keyword to specify that the table should be opened exclusively.

The columns list is a comma-separated list, enclosed in parentheses, of columns in the table and is optional. The VALUES clause is a comma-separated list of update values, enclosed in parentheses. Unless the source of new rows is a SELECT subquery, the VALUES clause is required and the number of update values in the VALUES clause must match the number of columns in the columns list exactly.

If no columns list is specified, incoming update values are stored in fields as they are defined sequentially in the table structure. Update values are applied to columns in the order the update values are listed in the VALUES clause. The number of update values must match the number of columns in the table exactly.

The following example inserts a single row into the Holdings table:

```
INSERT INTO Holdings
VALUES (4094095,'INPR',5000,10.500,'1998-01-02')
```

If an explicit columns list is stated, incoming update values (in the order they appear in the VALUES clause) are stored in the listed columns (in the order they appear in the columns list). NULL values are stored in any columns that are not in a columns list. When a columns list is explicitly described, there must be exactly the same number of update values in the VALUES clause as there are columns in the list.

The following example inserts a single row into the Customer table, adding data for only two of the columns in the table:

```
INSERT INTO "Customer" (CustNo, Company)
VALUES (9842,'Elevate Software, Inc.')
```

To add rows to one table that are retrieved from another table, omit the VALUES keyword and use a

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

subquery as the source for the new rows:

```
INSERT INTO "Customer" (CustNo, Company)
SELECT CustNo, Company
FROM "OldCustomer"
```

The INSERT statement only supports SELECT subqueries in the VALUES clause. References to tables other than the one to which rows are added or columns in such tables are only possible in SELECT subqueries.

The INSERT statement can use a single SELECT statement as the source for the new rows, but not multiple statements joined with UNION.

COMMIT Clause

The COMMIT clause is used to control how often DBISAM will commit a transaction while the INSERT statement is executing and/or whether the commit operation performs an operating system flush to disk. The INSERT statement implicitly uses a transaction if one is not already active. The default interval at which the implicit transaction is committed is based upon the record size of the table being updated in the query and the amount of buffer space available in DBISAM. The COMMIT INTERVAL clause is used to manually control the interval at which the transaction is committed based upon the number of rows inserted, and applies in both situations where a transaction was explicitly started by the application and where the transaction was implicitly started by DBISAM. In the case where a transaction was explicitly started by the application, the absence of a COMMIT INTERVAL clause in the SQL statement being executed will force DBISAM to never commit any of the effects of the SQL statement and leaves this up to the application to handle after the SQL statement completes. The syntax is as follows:

```
COMMIT [INTERVAL nnnn] [FLUSH]
```

The INTERVAL keyword is optional, allowing the application to use the default commit interval but still specify the FLUSH keyword to indicate that it wishes to have the transaction commits flushed to disk at the operating system level.

5.15 UPDATE Statement

Introduction

The SQL UPDATE statement is used to modify one or more existing rows in a table.

Syntax

```
UPDATE table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE]

SET column_ref = update_value
[, column_ref = update_value...]

[FROM table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE]

[[INNER | [LEFT | RIGHT] OUTER JOIN] table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE] ON join_condition]
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
[WHERE predicates]

[COMMIT [INTERVAL commit_interval] [FLUSH]]

[NOJOINOPTIMIZE]
[JOINOPTIMIZECOSTS]
```

Use the UPDATE statement to modify one or more column values in one or more existing rows in a single table per statement. Use a table reference in the UPDATE clause to specify the table to receive the data changes. Use the EXCLUSIVE keyword to specify that the table should be opened exclusively.

SET Clause

The SET clause is a comma-separated list of update expressions for the UPDATE statement. The syntax is as follows:

```
SET column_ref = update_value
[, column_ref = update_value...]
```

Each expression comprises the name of a column, the assignment operator (=), and the update value for that column. The update values in any one update expression may be literal values or calculated values.

FROM and JOIN Clauses

You may use an optional FROM clause with additional JOIN clauses to specify multiple tables from which an UPDATE statement retrieves data for the purpose of updating the target table. The value for a FROM clause is a comma-separated list of table names, with the first table exactly matching the table name specified after the UPDATE clause. Specified table names must follow DBISAM's SQL naming conventions for tables. Please see the Naming Conventions topic for more information. The following UPDATE statement updates data in one table based upon a LEFT OUTER JOIN condition to another table:

```
UPDATE orders SET ShipToContact=Customer.Contact
FROM orders LEFT OUTER JOIN customer
ON customer.custno=orders.custno
```

Note
The orders table must be specified twice - once after the UPDATE clause and again as the first table in the FROM clause.

You can use the AS keyword to specify a table correlation name, or alternately you can simply just specify the table correlation name after the source table name. The following example uses the second method to give each source table a shorter name to be used in qualifying source columns in the query:

```
UPDATE orders o SET ShipToContact=c.Contact
FROM orders o LEFT OUTER JOIN customer c
ON c.custno=o.custno
```

Use the EXCLUSIVE keyword to specify that the table should be opened exclusively.

Note
Be careful when using the EXCLUSIVE keyword with a table that is specified more than once in the same query, as is the case with recursive relationships between a table and itself.

The table reference cannot be passed to a FROM clause via a parameter. Please see the SELECT Statement topic for more information.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

WHERE Clause

The WHERE clause specifies filtering conditions for the UPDATE statement. The syntax is as follows:

```
WHERE predicates
```

Use a WHERE clause to limit the effect of a UPDATE statement to a subset of rows in the table, and the clause is optional.

The value for a WHERE clause is one or more logical expressions, or predicates, that evaluate to TRUE or FALSE for each row in the table. Only those rows where the predicates evaluate to TRUE are modified by an UPDATE statement. For example, the UPDATE statement below modifies all rows where the State column contains a value of 'CA':

```
UPDATE SalesInfo
SET TaxRate=0.0825
WHERE (State='CA')
```

Subqueries are supported in the WHERE clause. A subquery works like a search condition to restrict the number of rows updated by the outer, or "parent" query. Such subqueries must be valid SELECT statements. SELECT subqueries cannot be correlated in DBISAM's SQL, i.e. they cannot refer to columns in the outer (or "parent") statement.

Column correlation names cannot be used in filter comparisons in the WHERE clause. Use the actual column name.

Columns devoid of data contain NULL values. To filter using such column values, use the IS NULL predicate.

The UPDATE statement may reference any table that is specified in the UPDATE, FROM, or JOIN clauses in the WHERE clause.

COMMIT Clause

The COMMIT clause is used to control how often DBISAM will commit a transaction while the UPDATE statement is executing and/or whether the commit operation performs an operating system flush to disk. The UPDATE statement implicitly uses a transaction if one is not already active. The default interval at which the implicit transaction is committed is based upon the record size of the table being updated in the query and the amount of buffer space available in DBISAM. The COMMIT INTERVAL clause is used to manually control the interval at which the transaction is committed based upon the number of rows updated, and applies in both situations where a transaction was explicitly started by the application and where the transaction was implicitly started by DBISAM. In the case where a transaction was explicitly started by the application, the absence of a COMMIT INTERVAL clause in the SQL statement being executed will force DBISAM to never commit any of the effects of the SQL statement and leaves this up to the application to handle after the SQL statement completes. The syntax is as follows:

```
COMMIT [INTERVAL nnnn] [FLUSH]
```

The INTERVAL keyword is optional, allowing the application to use the default commit interval but still specify the FLUSH keyword to indicate that it wishes to have the transaction commits flushed to disk at the operating system level.

NOJOINOPTIMIZE Clause

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The NOJOINOPTIMIZE clause causes all join re-ordering to be turned off for a SELECT statement. The syntax is as follows:

```
NOJOINOPTIMIZE
```

Use a NOJOINOPTIMIZE clause to force the query optimizer to stop re-ordering joins for a SELECT statement. In certain rare cases the query optimizer might not have enough information to know that re-ordering the joins will result in worse performance than if the joins were left in their original order, so in such cases you can include this clause to force the query optimizer to not perform the join re-ordering.

JOINOPTIMIZECOSTS Clause

The JOINOPTIMIZECOSTS clause causes the optimizer to take into account I/O costs when optimizing join expressions. The syntax is as follows:

```
JOINOPTIMIZECOSTS
```

Use a JOINOPTIMIZECOSTS clause to force the query optimizer to use I/O cost projections to determine the most efficient way to process the conditions in a join expression. If you have a join expression with multiple conditions in it, then using this clause may help improve the performance of the join expression, especially if it is already executing very slowly.

5.16 DELETE Statement

Introduction

The SQL DELETE statement is used to delete one or more rows from a table.

Syntax

```
DELETE FROM table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE]

[[INNER | [LEFT | RIGHT] OUTER JOIN] table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE] ON join_condition]

[WHERE predicates]

[COMMIT [INTERVAL commit_interval] FLUSH]

[NOJOINOPTIMIZE]
[JOINOPTIMIZECOSTS]
```

Use DELETE to delete one or more rows from one existing table per statement.

FROM Clause

The FROM clause specifies the table to use for the DELETE statement. The syntax is as follows:

```
FROM table_reference
[AS correlation_name | correlation_name] [EXCLUSIVE]
```

Specified table names must follow DBISAM's SQL naming conventions for tables. Please see the Naming

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Conventions topic for more information.

Use the EXCLUSIVE keyword to specify that the table should be opened exclusively.

Note
Be careful when using the EXCLUSIVE keyword with a table that is specified more than once in the same query, as is the case with recursive relationships between a table and itself.

JOIN Clauses

You may use optional JOIN clauses to specify multiple tables from which a DELETE statement retrieves data for the purpose of deleting records in the target table. The following DELETE statement below deletes data in one table based upon an INNER JOIN condition to another table:

```
DELETE FROM orders
INNER JOIN customer ON customer.custno=orders.custno
WHERE customer.country='Bermuda'
```

You can use the AS keyword to specify a table correlation name, or alternately you can simply just specify the table correlation name after the source table name. The following example uses the second method to give each source table a shorter name to be used in qualifying source columns in the query:

```
DELETE FROM orders o
INNER JOIN customer c ON c.custno=o.custno
WHERE c.country='Bermuda'
```

Please see the SELECT Statement topic for more information.

WHERE Clause

The WHERE clause specifies filtering conditions for the DELETE statement. The syntax is as follows:

```
WHERE predicates
```

Use a WHERE clause to limit the effect of a DELETE statement to a subset of rows in the table, and the clause is optional.

The value for a WHERE clause is one or more logical expressions, or predicates, that evaluate to TRUE or FALSE for each row in the table. Only those rows where the predicates evaluate to TRUE are deleted by a DELETE statement. For example, the DELETE statement below deletes all rows where the State column contains a value of 'CA':

```
DELETE FROM SalesInfo
WHERE (State='CA')
```

Multiple predicates must be separated by one of the logical operators OR or AND. Each predicate can be negated with the NOT operator. Parentheses can be used to isolate logical comparisons and groups of comparisons to produce different row evaluation criteria.

Subqueries are supported in the WHERE clause. A subquery works like a search condition to restrict the number of rows deleted by the outer, or "parent" query. Such subqueries must be valid SELECT statements. SELECT subqueries cannot be correlated in DBISAM's SQL, i.e. they cannot refer to columns in the outer (or "parent") statement.

Column correlation names cannot be used in filter comparisons in the WHERE clause. Use the actual column name.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Columns devoid of data contain NULL values. To filter using such column values, use the IS NULL predicate.

The DELETE statement may reference any table that is specified in the FROM, or JOIN clauses in the WHERE clause.

COMMIT Clause

The COMMIT clause is used to control how often DBISAM will commit a transaction while the DELETE statement is executing and/or whether the commit operation performs an operating system flush to disk. The DELETE statement implicitly uses a transaction if one is not already active. The default interval at which the implicit transaction is committed is based upon the record size of the table being updated in the query and the amount of buffer space available in DBISAM. The COMMIT INTERVAL clause is used to manually control the interval at which the transaction is committed based upon the number of rows deleted, and applies in both situations where a transaction was explicitly started by the application and where the transaction was implicitly started by DBISAM. In the case where a transaction was explicitly started by the application, the absence of a COMMIT INTERVAL clause in the SQL statement being executed will force DBISAM to never commit any of the effects of the SQL statement and leaves this up to the application to handle after the SQL statement completes. The syntax is as follows:

```
COMMIT [INTERVAL nnnn] [FLUSH]
```

The INTERVAL keyword is optional, allowing the application to use the default commit interval but still specify the FLUSH keyword to indicate that it wishes to have the transaction commits flushed to disk at the operating system level.

NOJOINOPTIMIZE Clause

The NOJOINOPTIMIZE clause causes all join re-ordering to be turned off for a SELECT statement. The syntax is as follows:

```
NOJOINOPTIMIZE
```

Use a NOJOINOPTIMIZE clause to force the query optimizer to stop re-ordering joins for a SELECT statement. In certain rare cases the query optimizer might not have enough information to know that re-ordering the joins will result in worse performance than if the joins were left in their original order, so in such cases you can include this clause to force the query optimizer to not perform the join re-ordering.

JOINOPTIMIZECOSTS Clause

The JOINOPTIMIZECOSTS clause causes the optimizer to take into account I/O costs when optimizing join expressions. The syntax is as follows:

```
JOINOPTIMIZECOSTS
```

Use a JOINOPTIMIZECOSTS clause to force the query optimizer to use I/O cost projections to determine the most efficient way to process the conditions in a join expression. If you have a join expression with multiple conditions in it, then using this clause may help improve the performance of the join expression, especially if it is already executing very slowly.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

5.17 CREATE TABLE Statement

Introduction

The SQL CREATE TABLE statement is used to create a table.

Syntax

```
CREATE TABLE [IF NOT EXISTS] table_reference

(
column_name data type [dimensions]
[DESCRIPTION column description]
[NULLABLE][NOT NULL]
[DEFAULT default value]
[MIN | MINIMUM minimum value]
[MAX | MAXIMUM maximum value]
[CHARCASE UPPER | LOWER | NOCHANGE]
[COMPRESS 0..9]

[, column_name...]

[, [CONSTRAINT constraint_name]
[UNIQUE] [NOCASE]
PRIMARY KEY (column_name [[ASC |ASCENDING] | [DESC | DESCENDING]]
[, column_name...])
[COMPRESS DUPBYTE | TRAILBYTE | FULL | NONE]]

[TEXT INDEX (column_name, [column_name])]
[STOP WORDS space-separated list of words]
[SPACE CHARS list of characters]
[INCLUDE CHARS list of characters]

[DESCRIPTION table_description]

[INDEX PAGE SIZE index_page_size]
[BLOB BLOCK SIZE BLOB_block_size]

[LOCALE locale_name | LOCALE CODE locale_code]

[ENCRYPTED WITH password]

[USER MAJOR VERSION user-defined_major_version]
[USER MINOR VERSION user-defined_minor_version]

[LAST AUTOINC last_autoinc_value]
)
```

Use the CREATE TABLE statement to create a table, define its columns, and define a primary key constraint.

The specified table name must follow DBISAM's SQL naming conventions for tables. Please see the Naming Conventions topic for more information.

Column Definitions

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The syntax for defining a column is as follows:

```
column_name data type [dimensions]
[DESCRIPTION column description]
[NULLABLE][NOT NULL]
[DEFAULT default value]
[MIN or MINIMUM minimum value] [MAX or MAXIMUM maximum value]
[CHARCASE UPPER | LOWER | NOCHANGE]
[COMPRESS 0..9]
```

Column definitions consist of a comma-separated list of combinations of column name, data type and (if applicable) dimensions, and optionally their description, allowance of NULL values, default value, minimum and maximum values, character-casing, and compression level (for BLOB columns). The list of column definitions must be enclosed in parentheses. The number and type of dimensions that must be specified varies with column type.

DESCRIPTION Clause

The DESCRIPTION clause specifies the description for the column. The syntax is as follows:

```
DESCRIPTION column description
```

The description must be enclosed in single or double quotes and can be any value up to 100 characters in length.

NULLABLE and NOT NULL Clauses

The NULLABLE clause specifies that the column is not required and can be NULL. The NOT NULL clause specifies that the column is required and cannot be NULL. The syntax is as follows:

```
NULLABLE
NOT NULL
```

DEFAULT Clause

The DEFAULT clause specifies the default value for the column. The syntax is as follows:

```
DEFAULT default value
```

The default value must be a value that matches the data type of the column being defined. Also, the value must be expressed in ANSI/ISO format if it is a date, time, timestamp, or number. Please see the Naming Conventions topic for more information.

MINIMUM Clause

The MINIMUM clause specifies the minimum value for the column. The syntax is as follows:

```
MIN | MINIMUM minimum value
```

The minimum value must be a value that matches the data type of the column being defined. Also, the value must be expressed in ANSI/ISO format if it is a date, time, timestamp, or number. Please see the Naming Conventions topic for more information.

MAXIMUM Clause

The MAXIMUM clause specifies the maximum value for the column. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
MAX | MAXIMUM maximum value
```

The maximum value must be a value that matches the data type of the column being defined. Also, the value must be expressed in ANSI/ISO format if it is a date, time, timestamp, or number. Please see the Naming Conventions topic for more information.

CHARCASE Clause

The CHARCASE clause specifies the character-casing for the column. The syntax is as follows:

```
CHARCASE UPPER | LOWER | NOCHANGE
```

If the UPPER keyword is used, then all data values in this column will be upper-cased. If the LOWER keyword is used, then all data values in this column will be lower-cased. If the NOCHANGE keyword is used, then all data values for this column will be left in their original form. This clause only applies to string columns and is ignored for all others.

The following statement creates a table with columns that include descriptions and default values:

```
CREATE TABLE employee
(
  Last_Name CHAR(20) DESCRIPTION 'Last Name',
  First_Name CHAR(15) DESCRIPTION 'First Name',
  Hire_Date DATE DESCRIPTION 'Hire Date' DEFAULT CURRENT_DATE
  Salary NUMERIC(10,2) DESCRIPTION 'Salary' DEFAULT 0.00,
  Dept_No SMALLINT DESCRIPTION 'Dept #',
 PRIMARY KEY (Last_Name, First_Name)
)
```

Primary Index Definition

Use the PRIMARY KEY (or CONSTRAINT) clause to create a primary index for the new table. The syntax is as follows:

```
[, [CONSTRAINT constraint_name]
[UNIQUE] [NOCASE]
PRIMARY KEY (column_name [[ASC |ASCENDING] | [DESC | DESCENDING]]
[, column_name...])
[COMPRESS DUPBYTE | TRAILBYTE | FULL | NONE]]
```

The columns that make up the primary index must be specified. The UNIQUE flag is completely optional and is ignored since primary indexes are always unique. The alternate CONSTRAINT syntax is also completely optional and ignored.

A primary index definition can optionally specify that the index is case-insensitive and the compression used for the index.

NOCASE Clause

The NOCASE clause specifies the that the primary index should be sorted in case-insensitive order as opposed to the default of case-sensitive order. The syntax is as follows:

```
NOCASE
```

Columns Clause

The columns clause specifies a comma-separated list of columns that make up the primary index, and optionally whether the columns should be sorted in ascending (default) or descending order. The syntax

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

is as follows:

```
PRIMARY KEY (column_name [[ASC |ASCENDING] | [DESC | DESCENDING]]
[, column_name...])
```

The column names specified here must conform to the column naming conventions for DBISAM's SQL and must have been defined earlier in the CREATE TABLE statement. Please see the Naming Conventions topic for more information.

COMPRESS Clause

The COMPRESS clause specifies the type of index key compression to use for the primary index. The syntax is as follows:

```
COMPRESS DUPBYTE | TRAILBYTE | FULL | NONE
```

The DUPBYTE keyword specifies that duplicate-byte index key compression will be used, the TRAILBYTE keyword specifies that trailing-byte index key compression will be used, and the FULL keyword specifies that both duplicate-byte and trailing-byte index key compression will be used. The default index key compression is NONE.

The following statement creates a table with a primary index on the Last_Name and First_Name columns that is case-insensitive and uses full index key compression:

```
CREATE TABLE employee
(
  Last_Name CHAR(20) DESCRIPTION 'Last Name',
  First_Name CHAR(15) DESCRIPTION 'First Name',
  Hire_Date DATE DESCRIPTION 'Hire Date' DEFAULT CURRENT_DATE
  Salary NUMERIC(10,2) DESCRIPTION 'Salary' DEFAULT 0.00,
  Dept_No SMALLINT DESCRIPTION 'Dept #',
  NOCASE PRIMARY KEY (Last_Name, First_Name) COMPRESS FULL
)
```

Note
Primary indexes are the only form of constraint that can be defined with CREATE TABLE.

Full Text Indexes Definitions

Use the TEXT INDEX, STOP WORDS, SPACE CHARS, and INCLUDE CHARS clauses (in that order) to create a full text indexes for the new table. The syntax is as follows:

```
TEXT INDEX (column_name, [column_name])
STOP WORDS space-separated list of words
SPACE CHARS list of characters
INCLUDE CHARS list of characters
```

When using full text indexing, the TEXT INDEX clause is required and consists of a comma-separated list of columns that should be full text indexed. The column names specified here must conform to the column naming conventions for DBISAM's SQL and must have been defined earlier in the CREATE TABLE statement. Please see the Naming Conventions topic for more information.

The STOP WORDS clause is optional and consists of a space-separated list of words as a string that specify the stop words used for the full text indexes.

The SPACE CHARS and INCLUDE CHARS clauses are optional and consist of a set of characters as a string that specify the space and include characters used for the full text indexes.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Table Description

Use the DESCRIPTION clause to specify a description for the table. The syntax is as follows:

```
DESCRIPTION table_description
```

The description is optional and should be specified as a string.

Table Index Page Size

Use the INDEX PAGE SIZE clause to specify the index page size for the table. The syntax is as follows:

```
INDEX PAGE SIZE index_page_size
```

The index page size is optional and should be specified as an integer in the 1024 bytes to 16 kilobytes range.

Table BLOB Block Size

Use the BLOB BLOCK SIZE clause to specify the BLOB block size for the table. The syntax is as follows:

```
BLOB BLOCK SIZE BLOB_block_size
```

The BLOB block size is optional and should be specified as an integer in the 64 bytes to 64 kilobytes range.

Table Locale

Use the LOCALE clause to specify the locale for the table. The syntax is as follows:

```
LOCALE locale_name | LOCALE CODE locale_code
```

The locale is optional and should be specified as an identifier enclosed in double quotes ("") or square brackets ([]) if specifying a locale constant, or as an integer value if specifying a locale ID. If this clause is not specified, then the default "ANSI Standard" locale (ID 0) will be used for the table.

Table Encryption

Use the ENCRYPTED WITH clause to specify whether the table should be encrypted with a password. The syntax is as follows:

```
ENCRYPTED WITH password
```

Table encryption is optional and the password for this clause should be specified as a string constant enclosed in single quotes ('').

User-Defined Versions

Use the USER MAJOR VERSION and USER MINOR VERSION clauses to specify user-defined version numbers for the table. The syntax is as follows:

```
USER MAJOR VERSION user-defined_major_version
[USER MINOR VERSION user-defined_minor_version]
```

User-defined versions are optional and the versions should be specified as integers.

Last Autoinc Value

Use the LAST AUTOINC clause to specify the last autoinc value for the table. The syntax is as follows:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
LAST AUTOINC last_autoinc_value
```

The last autoinc value is optional and should be specified as an integer. If this clause is not specified, the default last autoinc value is 0.

5.18 CREATE INDEX Statement

Introduction

The SQL CREATE INDEX statement is used to create a secondary index for a table.

Syntax

```
CREATE [UNIQUE] [NOCASE]
INDEX [IF NOT EXISTS] index_name

ON table_reference

(column_name [ASC or ASCENDING | DESC or DESCENDING]
[, column_name...])
[COMPRESS DUPBYTE | TRAILBYTE | FULL | NONE]]
```

Use the CREATE INDEX statement to create a secondary index for an existing table. If index names contain embedded spaces they must be enclosed in double quotes ("") or square brackets ([]). Secondary indexes may be based on multiple columns.

UNIQUE Clause

Use the UNIQUE clause to create an index that raises an error if rows with duplicate column values are inserted. By default, indexes are not unique. The syntax is as follows:

```
UNIQUE
```

NOCASE Clause

The NOCASE clause specifies the that the secondary index should be sorted in case-insensitive order as opposed to the default of case-sensitive order. The syntax is as follows:

```
NOCASE
```

Columns Clause

The columns clause specifies a comma-separated list of columns that make up the secondary index, and optionally whether the columns should be sorted in ascending (default) or descending order. The syntax is as follows:

```
(column_name [[ASC |ASCENDING] | [DESC | DESCENDING]]
[, column_name...])
```

The column names specified here must conform to the column naming conventions for DBISAM's SQL and must have been defined earlier in the CREATE TABLE statement. Please see the Naming Conventions topic for more information.

COMPRESS Clause

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The COMPRESS clause specifies the type of index key compression to use for the secondary index. The syntax is as follows:

```
COMPRESS DUPBYTE | TRAILBYTE | FULL | NONE
```

The DUPBYTE keyword specifies that duplicate-byte index key compression will be used, the TRAILBYTE keyword specifies that trailing-byte index key compression will be used, and the FULL keyword specifies that both duplicate-byte and trailing-byte index key compression will be used. The default index key compression is NONE.

The following statement creates a multi-column secondary index that sorts in ascending order for the CustNo column and descending order for the SaleDate column:

```
CREATE INDEX CustDate
ON Orders (CustNo, SaleDate DESC) COMPRESS DUPBYTE
```

The following statement creates a unique, case-insensitive secondary index:

```
CREATE UNIQUE NOCASE INDEX "Last Name"
ON Employee (Last_Name) COMPRESS FULL
```

5.19 ALTER TABLE Statement

Introduction

The SQL ALTER TABLE statement is used to restructure a table.

Syntax

```
ALTER TABLE [IF EXISTS] table_reference

[[ADD [COLUMN] [IF NOT EXISTS]
column_name data type [dimensions]
[AT column_position]
[DESCRIPTION column description]
[NULLABLE][NOT NULL]
[DEFAULT default value]
[MIN or MINIMUM minimum value]
[MAX or MAXIMUM maximum value]
[CHARCASE UPPER | LOWER | NOCHANGE]
[COMPRESS 0..9]]

|

[REDEFINE [COLUMN] [IF EXISTS]
column_name [new_column_name] data type [dimensions]
[AT column_position]
[DESCRIPTION column description]
[NULLABLE][NOT NULL]
[DEFAULT default value]
[MIN or MINIMUM minimum value]
[MAX or MAXIMUM maximum value]
[CHARCASE UPPER | LOWER | NOCHANGE]
[COMPRESS 0..9]]
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
|

[DROP [COLUMN] [IF EXISTS] column_name]]

[, ADD [COLUMN] column_name
REDEFINE [COLUMN] column_name
DROP [COLUMN] column_name...]

[, ADD [CONSTRAINT constraint_name]
[UNIQUE] [NOCASE] PRIMARY KEY
(column_name [ASC or ASCENDING | DESC or DESCENDING]
[, column_name...])
[COMPRESS DUPBYTE | TRAILBYTE | FULL | NONE]]

[, REDEFINE [CONSTRAINT constraint_name]
[UNIQUE] [NOCASE] PRIMARY KEY
(column_name [ASC or ASCENDING | DESC or DESCENDING]
[, column_name...])
[COMPRESS DUPBYTE | TRAILBYTE | FULL | NONE]]

[, DROP [CONSTRAINT constraint_name] PRIMARY KEY]

[TEXT INDEX (column_name, [column_name])]
[STOP WORDS space-separated list of words]
[SPACE CHARS list of characters]
[INCLUDE CHARS list of characters]

[DESCRIPTION table_description]

[INDEX PAGE SIZE index_page_size]
[BLOB BLOCK SIZE BLOB_block_size]

[LOCALE locale_name | LOCALE CODE locale_code]

[ENCRYPTED WITH password]

[USER MAJOR VERSION user-defined_major_version]
[USER MINOR VERSION user-defined_minor_version]

[LAST AUTOINC last_autoinc_value]

[NOBACKUP]
```

Use the ALTER TABLE statement to alter the structure of an existing table. It is possible to delete one column and add another in the same ALTER TABLE statement as well as redefine an existing column without having to first drop the column and then re-add the same column name. This is what is sometimes required with other database engines and can result in loss of data. DBISAM's REDEFINE keyword removes this problem. In addition, the IF EXISTS and IF NOT EXISTS clauses can be used with the ADD, REDEFINE, and DROP keywords to allow for action on columns only if they do or do not exist.

The DROP keyword requires only the name of the column to be deleted. The ADD keyword requires the same combination of column name, data type and possibly dimensions, and extended column definition information as the CREATE TABLE statement when defining new columns.

The statement below deletes the column FullName and adds the column LastName, but only if the LastName column doesn't already exist:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
ALTER TABLE Names
DROP FullName,
ADD IF NOT EXISTS LastName CHAR(25)
```

It is possible to delete and add a column of the same name in the same ALTER TABLE statement, however any data in the column is lost in the process. An easier way is to use the extended syntax provided by DBISAM's SQL with the REDEFINE keyword:

```
ALTER TABLE Names
REDEFINE LastName CHAR(30)
```

Note
In order to remove the full text index completely, you would specify no columns in the TEXT INDEX clause like this:

```
ALTER TABLE Customer
TEXT INDEX ()
```

NOBACKUP Clause

The NOBACKUP clause specifies that no backup files should be created during the process of altering the table's structure.

Please see the CREATE TABLE statement for more information on all other clauses used in the ALTER TABLE statement. Their usage is the same as with the CREATE TABLE statement.

5.20 EMPTY TABLE Statement

Introduction

The SQL EMPTY TABLE statement is used to empty a table of all data.

Syntax

```
EMPTY TABLE [IF EXISTS] table_reference
```

Use the EMPTY TABLE statement to remove all data from an existing table. The statement below empties a table:

```
EMPTY TABLE Employee
```

5.21 OPTIMIZE TABLE Statement

Introduction

The SQL OPTIMIZE TABLE statement is used to optimize a table.

Syntax

```
OPTIMIZE TABLE [IF EXISTS] table_reference
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
[ON index_name]

[NOBACKUP]
```

Use the OPTIMIZE TABLE statement to remove all free space from a table and organize the data more efficiently.

ON Clause

The ON clause is optional and specifies the name of an index in the table to use for organizing the physical data records. It is usually recommended that you do not specify this clause, which will result in the table being organized using the primary index.

NOBACKUP Clause

The NOBACKUP clause specifies that no backup files should be created during the process of optimizing the table.

The statement below optimizes a table and suppresses any backup files:

```
OPTIMIZE TABLE Employee NOBACKUP
```

5.22 EXPORT TABLE Statement

Introduction

The SQL EXPORT TABLE statement is used to export a table to a delimited text file.

Syntax

```
EXPORT TABLE [IF EXISTS] table_reference [EXCLUSIVE]

TO text_file_name

[DELIMITER delimiter_character]

[WITH HEADERS]

[COLUMNS (column_name [, column_name])]

[DATE date_format]
[TIME time_format]
[DECIMAL decimal_separator]
```

Use the EXPORT TABLE statement to export a table to a delimited text file specified by the TO clause. The file name must be enclosed in double quotes ("") or square brackets ([]) if it contains a drive, path, or file extension. Use the EXCLUSIVE keyword to specify that the table should be opened exclusively.

DELIMITER Clause

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The DELIMITER clause is optional and specifies the delimiter character to use in the exported text file. The DELIMITER character should be specified as a single character constant enclosed in single quotes ('') or specified using the pound (#) sign and the ASCII character value. The default delimiter character is the comma (,).

WITH HEADERS Clause

The WITH HEADERS clause is optional and specifies that the exported text file should contain column headers for all columns as the first row.

COLUMNS Clause

The columns clause is optional and specifies a comma-separated list of columns that should be exported to the text file. The column names specified here must conform to the column naming conventions for DBISAM's SQL and must exist in the table being exported. Please see the Naming Conventions topic for more information.

DATE, TIME, and DECIMAL Clauses

The DATE, TIME, and DECIMAL clauses are optional and specify the formats and decimal separator that should be used when exporting dates, times, timestamps, and numbers. The DATE and TIME formats should be specified as string constants enclosed in single quotes ('') and the DECIMAL separator should be specified as a single character constant enclosed in single quotes ('') or specified using the pound (#) sign and the ASCII character value. The default date format is 'yyyy-mm-dd', the default time format is 'hh:mm:ss.zzz ampm', and the default decimal separator is '.'.

The statement below exports three fields from the Employee table into a file called 'employee.txt':

```
EXPORT TABLE Employee
TO "c:\mydata\employee.txt"
WITH HEADERS
COLUMNS (ID, FirstName, LastName)
```

5.23 IMPORT TABLE Statement

Introduction

The SQL IMPORT TABLE statement is used to import data from delimited text file into a table.

Syntax

```
IMPORT TABLE [IF EXISTS] table_reference

FROM text_file_name

[DELIMITER delimiter_character]

[WITH HEADERS]

[COLUMNS (column_name [, column_name])]
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
[DATE date_format]
[TIME time_format]
[DECIMAL decimal_separator]
```

Use the IMPORT TABLE statement to import data into a table from a delimited text file specified by the FROM clause. The file name must be enclosed in double quotes ("") or square brackets ([]) if it contains a drive, path, or file extension. Use the EXCLUSIVE keyword to specify that the table should be opened exclusively.

DELIMITER Clause

The DELIMITER clause is optional and specifies the delimiter character used in the imported text file to separate data from different columns. The DELIMITER character should be specified as a single character constant enclosed in single quotes ('') or specified using the pound (#) sign and the ASCII character value. The default delimiter character is the comma (,).

WITH HEADERS Clause

The WITH HEADERS clause is optional and specifies that the imported text file contains column headers for all columns as the first row. In such a case DBISAM will not import this row as a record but will instead ignore it.

COLUMNS Clause

The columns clause is optional and specifies a comma-separated list of columns that the imported text file contains. If the imported text file contains column data in a different order than that of the table, or only a subset of column data, then it is very important that this clause be used. Also, the column names specified here must conform to the column naming conventions for DBISAM's SQL and must exist in the table being exported. Please see the Naming Conventions topic for more information.

DATE, TIME, and DECIMAL Clauses

The DATE, TIME, and DECIMAL clauses are optional and specify the formats and decimal separator that should be used when importing dates, times, timestamps, and numbers from the text file. The DATE and TIME formats should be specified as string constants enclosed in single quotes ('') and the DECIMAL separator should be specified as a single character constant enclosed in single quotes ('') or specified using the pound (#) sign and the ASCII character value. The default date format is 'yyyy-mm-dd', the default time format is 'hh:mm:ss.zzz ampm', and the default decimal separator is '.'.

The statement below imports three fields from a file called 'employee.txt' into the Employee table:

```
IMPORT TABLE Employee
FROM "c:\mydata\employee.txt"
WITH HEADERS
COLUMNS (ID, FirstName, LastName)
```

5.24 REPAIR TABLE Statement

Introduction

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

The SQL REPAIR TABLE statement is used to repair a table that is corrupted or suspected of being corrupted.

Syntax

```
REPAIR TABLE [IF EXISTS] table_reference

FORCEINDEXREBUILD
```

Use the REPAIR TABLE statement to repair the physical structure of a table that is corrupted or suspected of being corrupted.

FORCEINDEXREBUILD Clause

Use the FORCEINDEXREBUILD clause to force the indexes in the table to be rebuilt regardless of whether they are determined to be corrupted or not. Sometimes there is corruption in indexes that DBISAM cannot detect in the table verification or repair process, and this clause will resolve such an issue.

The statement below repairs a table:

```
REPAIR TABLE Employee
```

5.25 UPGRADE TABLE Statement

Introduction

The SQL UPGRADE TABLE statement is used to upgrade a table from a previous DBISAM table format to the current table format.

Syntax

```
UPGRADE TABLE [IF EXISTS] table_reference
```

Use the UPGRADE TABLE statement to upgrade a table to the current DBISAM table format. The statement below upgrades a table:

```
UPGRADE TABLE Employee
```

5.26 DROP TABLE Statement

Introduction

The SQL DROP TABLE statement is used to delete a table.

Syntax

```
DROP TABLE [IF EXISTS] table_reference
```

Use the DROP TABLE statement to delete an existing table. The statement below drops a table:

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
DROP TABLE Employee
```

5.27 RENAME TABLE Statement

Introduction

The SQL RENAME TABLE statement is used to rename a table.

Syntax

```
RENAME TABLE [IF EXISTS] table_reference
TO table_reference
```

Use the RENAME TABLE statement to rename a table. The statement below renames a table:

```
RENAME TABLE Employee
TO Employees
```

5.28 DROP INDEX Statement

Introduction

The SQL DROP INDEX statement is used to delete a primary or secondary index from a table.

Syntax

```
DROP INDEX [IF EXISTS]
table_reference.index_name | PRIMARY
```

Use the DROP INDEX statement to delete a primary or secondary index. To delete a secondary index, identify the index using the table name and index name separated by an identifier connector symbol (.):

```
DROP INDEX Employee."Last Name"
```

To delete a primary index, identify the index with the keyword PRIMARY:

```
DROP INDEX Orders.PRIMARY
```

5.29 START TRANSACTION Statement

Introduction

The SQL START TRANSACTION statement is used to start a transaction on the current database.

Syntax

```
START TRANSACTION
```

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

```
[WITH <comma-separated list of tables>]
```

Use the START TRANSACTION statement to start a transaction. The WITH clause allows to start a restricted transaction on a specified set of tables. It accepts a comma-delimited list of table names to include in the restricted transaction.

5.30 COMMIT Statement

Introduction

The SQL COMMIT statement is used to commit an active transaction on the current database.

Syntax

```
COMMIT [WORK] [FLUSH]
```

Use the COMMIT statement to commit an active transaction. You may optionally include the WORK keyword for compatibility with the SQL standard.

FLUSH Clause

Use the FLUSH clause to indicate that the commit operation should also instruct the operating system to flush all committed data to disk.

5.31 ROLLBACK Statement

Introduction

The SQL ROLLBACK statement is used to rollback an active transaction on the current database.

Syntax

```
ROLLBACK [WORK]
```

Use the ROLLBACK statement to rollback an active transaction. You may optionally include the WORK keyword for compatibility with the SQL standard.

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Index

- - -

- arithmetic operator in SQL 240
- date/time operator in SQL 240

- & -

& accelerator character 194

- * -

* arithmetic operator in SQL 240

- / -

/ arithmetic operator in SQL 240

- | -

|| string concatenation operator in SQL 240

- + -

+ arithmetic operator in SQL 240
+ date/time operator in SQL 240
+ string concatenation operator in SQL 240

- < -

< SQL comparison operator 240
<= SQL comparison operator 240
<> SQL comparison operator 240

- = -

= SQL comparison operator 240

- > -

> SQL comparison operator 240
>= SQL comparison operator 240

- A -

ABS numeric function in SQL 254
Accelerator characters 194
Access - import from 140
Access privileges 97
Access privileges introduction 94
ACOS numeric function in SQL 254
ADD clause in SQL ALTER TABLE 295
Add Database command 9
Administration 97
Administration button 89
Administrative Hidden/Sticky 84
Administrative Styles 81
Admin-Style Mode toggle button 81
Aggregate functions in SQL 261
Align Bottoms button 163
Align Centers (H) button 163
Align Centers (V) button 163
Align Left paragraph alignment button 80
Align Left Sides button 163
Align Right paragraph alignment button 80
Align Right Sides button 163
Align Text buttons 186
Align Tops button 163
Alignment Lines 163
Alignment of items 175
Alignment of paragraphs 36
ALL clause in SQL SELECT 271
Alphabets 221
ALTER TABLE statement in SQL 295
Alternative Synopses 87
Alventis introduction 4
Alventis overview for the experts 2
Ampersand accelerator character 194
Anchor Mode 175
Anchors 175
AND logical operator in SQL 240
AND search operator 131
ANSI SQL-92 support overview 230
Application Settings for Alventis 133
Arbitration 81
Arithmetic operators in SQL 240
AS clause in SQL SELECT 271
ASCENDING clause in SQL ALTER TABLE 295
ASCENDING clause in SQL CREATE INDEX 294

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

ASCENDING clause in SQL CREATE TABLE 289
Ascending sort order 22
ASCII code of symbols 47
ASIN numeric function in SQL 254
AT clause in SQL ALTER TABLE 295
ATAN numeric function in SQL 254
ATAN2 numeric function in SQL 254
Attribute inheritance 186
Authentication Dialog 113
Author System field 208
AutoCorrect 75
Auto-Expand in Report checkbox 118
AutoInc functions in SQL 265
Auto-Increment data type 154
AutoSize item attribute 186
Auto-snap to Gridlines toggle button 163
Average summary type 28
AVG aggregate function in SQL 261

- B -

Background Color button 39
Background of Panels 186
Backup 223
Banded reports 118
Bar customization 214
Bar style settings 133
BBK 223
BETWEEN SQL extended comparison operator 240
Bitmap 59
BLB 223
BLOB BLOCK SIZE clause in SQL ALTER TABLE 295
BLOB BLOCK SIZE clause in SQL CREATE TABLE 289
BLOB field storage 223
BMP 59
Bold button 80
Boolean data type 154
Boolean functions in SQL 259
Boolean logic 22
Border Color InstaButton 186
Border Style InstaButton 186
Borders of Memo tables 50
Borders of paragraphs 36
Both Colors button 39
Bring to Front button 163
Browsers and non-Latin text Cut-and-Paste 221
Bullets and Numbering InstaButton 41

- C -

Caching of credentials 108
Calculated Fields 197
Calculator Boxes 19
Calculator Edit Box item introduced 154
Cancel DataNav button 16
Caption editing 172
CASE value operator in SQL 240
CAST function in SQL 267
Category System field 208
CEIL numeric function in SQL 254
CEILING numeric function in SQL 254
Cell focus 22
Cell Format InstaButton 50
Cells of Memo tables 50
Center 186
Center in Parent (H) button 163
Center in Parent (V) button 163
Center paragraph alignment button 80
Change Password button 89
Character Scale 33
Character set 221
Character Sets 33
Character Spacing 33
CHARCASE clause in SQL ALTER TABLE 295
CHARCASE clause in SQL CREATE TABLE 289
Charset 221
Charset in Font dialog 33
Charset Priority button 33
Checkbox caption editing 172
Checkbox item introduced 154
Checkbox Null Style 186
Child relationship 175
Clear a Single Match button 66
Clear All Matches button 66
Clear Bullets/Numbering button 41
Clear Cell Formatting button 50
Clear Sticky/Hidden button 84
Clear Style button in Designer 186
Clear Styles button 80
Client Alignment 175
Client-Server 89
Clipboard operations between Containers 163
Clone UniGrid command 9

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Close Filter button 22
COALESCE boolean function in SQL 259
Collapsing grouping row 22
Color Picker 39
COLUMN clause in SQL ALTER TABLE 295
Column correlation names 231
Column header 22
Column hiding 22
Column moving 22
Column naming conventions in SQL 231
Column sizing 22
COLUMNS clause in SQL EXPORT TABLE 298
COLUMNS clause in SQL IMPORT TABLE 299
Columns of tables defined 208
COMMIT clause in SQL INSERT 282
COMMIT statement in SQL 303
Comparison operators in SQL 240
COMPRESS clause in SQL ALTER TABLE 295
COMPRESS clause in SQL CREATE INDEX 294
COMPRESS clause in SQL CREATE TABLE 289
CONCAT string function in SQL 248
Constant functions in SQL 270
CONSTRAINT clause in SQL ALTER TABLE 295
CONSTRAINT clause in SQL CREATE TABLE 289
Containers 175
Controls on InfoViews 154
Convert Link to Text button 62
Copy-and-Paste between Containers 163
Corruption of data 223
COS numeric function in SQL 254
COT numeric function in SQL 254
COUNT aggregate function in SQL 261
Count summary type 28
CREATE INDEX statement in SQL 294
CREATE TABLE statement in SQL 289
Create/Reopen UniGrid button 9
CreationDate System field 208
Credentials Caching 108
CSV - import from 140
Currency data type 154
Currency Edit Box item introduced 154
Currency Precision 186
CURRENT_DATE function in SQL 270
CURRENT_GUID function in SQL 270
CURRENT_TIME function in SQL 270
CURRENT_TIMESTAMP function in SQL 270
CURRENT_USER function in SQL 270
Custom colors palette 39
Custom Filter dialog 22
Custom Gridlines button 163
Customization dialog 22
Customize button 22
Customize dialog (bars/menus) 214
Customizing toolbars and menus 214
Cut-and-Paste between Containers 163
Cut-and-Paste non-Latin text from web browsers 221

- D -

DAT 223
Data backup 223
Data Conversion functions in SQL 267
Data Import/Export 140
Data Types 154
Data verification and repair 223
Database Explorer 89
Database manipulation 89
Database principles 208
Databases grid 89
Data-entry forms introduction 4
DataNav toolbar 16
Date Boxes 19
DATE clause in SQL EXPORT TABLE 298
DATE clause in SQL IMPORT TABLE 299
Date constants in SQL 231
Date data type 154
Date Edit Box item introduced 154
Date operators in SQL 240
Date/Time data type 154
Date/Time inserting 80
DAYSFROMMSECS function in SQL 267
DBF - import from 140
DBISAM Database Server 205
DBISAM introduced 230
DBK 223
DBMS - differences from 4
dbsrvr 205
DECIMAL clause in SQL EXPORT TABLE 298
DECIMAL clause in SQL IMPORT TABLE 299
DEFAULT clause in SQL ALTER TABLE 295
DEFAULT clause in SQL CREATE TABLE 289
Default queries 114
DEGREES numeric function in SQL 254
Delete Data Table button 9

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Delete DataNav button 16
Delete Selected Columns button 50
Delete Selected Rows button 50
DELETE statement in SQL 286
Delimited files import/export 140
DELIMITER clause in SQL EXPORT TABLE 298
DELIMITER clause in SQL IMPORT TABLE 299
Deployment of the Server 205
DESCENDING clause in SQL ALTER TABLE 295
DESCENDING clause in SQL CREATE INDEX 294
DESCENDING clause in SQL CREATE TABLE 289
Descending sort order 22
DESCRIPTION clause in SQL ALTER TABLE 295
DESCRIPTION clause in SQL CREATE TABLE 289
Design Item Hints toggle 163
Design Mode 163
Designer 152
Designer introduction 4
Designer overview for the experts 2
Detail InfoView 16
Dictionaries 75
DISTINCT clause in SQL SELECT 271
Doorstop item 175
Drag-and-Drop in Memos 79
DROP clause in SQL ALTER TABLE 295
DROP INDEX statement in SQL 302
DROP TABLE statement in SQL 301
DUPBYTE COMPRESS clause in SQL ALTER TABLE 295
DUPBYTE COMPRESS clause in SQL CREATE INDEX 294
DUPBYTE COMPRESS clause in SQL CREATE TABLE 289
Dynamic grid Summaries 28

- E -

Edit Box item introduced 154
Editing grid records 22
Embedded comments in SQL 231
EMF 59
EMPTY TABLE statement in SQL 297
Enable GIF Animation button 59
Enabled item attribute 186
ENCRYPTED WITH clause in SQL ALTER TABLE 295
ENCRYPTED WITH clause in SQL CREATE TABLE 289
ENCRYPTED WITH clause in SQL SELECT 271
Encryption of tables 97
Enhanced Metafiles 59
Event Log of servers 108
Excel - import from 140
EXCEPT clause in SQL SELECT 271
EXCLUSIVE clause in SQL SELECT 271
EXP numeric function in SQL 254
Expand Height to Tallest button 163
Expand Width to Widest button 163
Expanding grouping row 22
Export of Data 140
Export of InfoSets 134
EXPORT TABLE statement in SQL 298
Extended comparison operators in SQL 240
EXTRACT function in SQL 267

- F -

Features overview 2
Field creation/deletion/modification 154
Field Types 154
Fields Grid in FieldViews 154
Fields of tables defined 208
FieldView 154
FieldViews button 154
File Hyperlinks 62
Files of a table 223
Files of Alventis 223
Filter Box 22
Filter Builder dialog 22
Filter button 22
Filter criteria 22
Filtering expression 22
Filtering records 22
Find and Replace 72
Find in Memos 72
Find Next in Memo button 72
First DataNav button 16
Fixed width table 50
Fixed-width Text files - import from 140
Float data type 154
FLOOR numeric function in SQL 254
FLUSH clause in SQL COMMIT 303
FLUSH clause in SQL INSERT 282
Focusing cells 22

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Font Charset 221
Font Color button 39
Font combo box 80
Font dropdown dialog 33
Font InstaButton 33
Font Priority button 33
Footer Summary 28
Footers of reports 118
FORCEINDEXREBUILD clause in SQL REPAIR TABLE 300
Foreign keys 208
Forget Password button 89
Format Painter 80
Formatting InfoView Items 186
Formula 197
FROM clause in SQL IMPORT TABLE 299
FROM clause in SQL SELECT 271
FULL COMPRESS clause in SQL ALTER TABLE 295
FULL COMPRESS clause in SQL CREATE INDEX 294
FULL COMPRESS clause in SQL CREATE TABLE 289
Full Text Indexing functions in SQL 265
Full-text index table 223
Full-Text Indexing 9

- G -

GIF 59
Global Match Highlight Level button 66
Graphic data type 154
Grayed-Out State 9
Greened-In State 9
Grid 22
Grid button 170
Grid cell 22
Grid Cell Auto Height toggle button 22
Grid column 22
Grid Column Caption editing 172
Grid column creation/deletion/selection/manipulation 170
Grid Copy-and-Paste on InfoViews 163
Grid creation on InfoViews 170
Grid footer Summary 28
Grid group Summary 28
Grid InfoView 16
Grid Item selection combo box 186
Grid record 22
Grid row 22
Gridline Step 163
Gridlines 163
Group by Box 22
GROUP BY clause in SQL SELECT 271
Group Summary 28
Grouping records 22
Groups 97
Groups introduction 94

- H -

HAVING clause in SQL SELECT 271
Header Sorting Arrow 22
Headers of reports 118
Heavyweight items 163
Height SpinEdit for Item sizing 163
Help organization 4
Hidden records 84
Hiding columns 22
Highlighted record 22
Highlighting in Memos 66
Hints for controls and Memos 133
Hotkey customization 214
HotTrack item attribute 186
HOURSFROMMSECS function in SQL 267
Hover hyperlink text color 62
HTML - import/export 149
Hyperlink Caption 62
Hyperlink Target 62
Hyperlinks 62

- I -

IBK 223
Icons (ICO) 59
ID System field 208
IDENT_CURRENT autoinc function in SQL 265
IDW 223
IDX 223
IF boolean function in SQL 259
IF EXISTS clause in SQL ALTER TABLE 295
IF EXISTS clause in SQL EMPTY TABLE 297
IF EXISTS clause in SQL OPTIMIZE TABLE 297
IF NOT EXISTS clause in SQL ALTER TABLE 295
IF NOT EXISTS clause in SQL CREATE INDEX 294

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

IF NOT EXISTS clause in SQL CREATE TABLE 289
IFNULL boolean function in SQL 259
Ifxs system tables 223
Image (static) creation 172
Image Boxes 19
Image Format 59
Image item introduced 154
Images - inserting in Memos 59
Immediate Lookup Combo editing 172
Immediate Lookup Combo item introduced 154
Import Modes 134
Import of Data 140
Import of InfoSets 134
IMPORT TABLE statement in SQL 299
IN SQL extended comparison operator 240
INCLUDE CHARS clause in SQL ALTER TABLE 295
INCLUDE CHARS clause in SQL CREATE TABLE 289
Indent button 80
Indents of paragraphs 36
Index corruption and repair 223
INDEX PAGE SIZE clause in SQL ALTER TABLE 295
INDEX PAGE SIZE clause in SQL CREATE TABLE 289
InfoSet 134
InfoSet Import/Export 134
InfoView 16
InfoView creation/deletion 160
InfoView Grid in FieldViews 160
InfoView introduction 4
InfoView Item creation 161
InfoView Item manipulation 163
InfoView Item moving/resizing 163
InfoView Items 154
InfoView Items Overview 172
InfoView opening 9
InfoView relationship with Tables 160
InfoView Title/Caption 160
Inheritance of attributes 186
INNER clause in SQL SELECT 271
Insert Column on the Left button 50
Insert Column on the Right button 50
Insert DataNav button 16
Insert Date/Time buttons 80
Insert Link to File button 62
Insert Link to Record button 62
Insert Page Break button 80
Insert Picture button 59
Insert Row Above button 50
Insert Row Below button 50
INSERT statement in SQL 282
Insert Table InstaButton 50
Insert/Edit Hyperlink button 62
InstaButton multiplicity 214
InstaButtons brief introduction 30
InstaButtons overview for the experts 2
Installation of the Server 205
InstaSearch mode 9
Integer data type 154
Interface persistence 19
International issues 221
Internet Hyperlinks 62
INTERSECT clause in SQL SELECT 271
INTERVAL clause in SQL INSERT 282
INTO clause in SQL INSERT 282
INTO clause in SQL SELECT 271
Introduction to Alventis and Designer 4
Introduction to Alventis for the experts 2
IS NULL SQL extended comparison operator 240
Italic button 80
Item Formatting 186
Item Hints 163
Item manipulation on InfoViews 163
Items on InfoViews 154
Items on InfoViews Overview 172

- J -

JOIN clause in SQL SELECT 271
JOINOPTIMIZECOSTS clause in SQL SELECT 271
JPEG 59
JPG 59
Justify paragraph alignment button 80

- K -

Keep lines together 36
Keep paragraph with next 36
Keep Table Together button 50
Keyboard shortcuts 214
KeyMap 214
Kill UniGrid command 9

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

- L -

Label caption editing 172
Label creation 172
Languages and internationalization 221
Large Integer data type 154
LAST AUTOINC clause in SQL ALTER TABLE 295
LAST AUTOINC clause in SQL CREATE TABLE 289
Last DataNav button 16
LASTAUTOINC autoinc function in SQL 265
LastModificationDate System field 208
Layout of InfoViews 175
Layout Templates 172
LCASE string function in SQL 248
LEFT clause in SQL SELECT 271
LEFT string function in SQL 248
Legal bullets and numbering 41
Length of fields 154
LENGTH string function in SQL 248
Lengths of String fields 208
Lightweight items 163
LIKE search operator 131
LIKE SQL extended comparison operator 240
Limits and system capacities 227
Line spacing 36
Linked tables 199
Lists (bulleted/numbered) 41
LiveSpell 75
Local mode 89
LOCALE clause in SQL ALTER TABLE 295
LOCALE clause in SQL CREATE TABLE 289
LOCALE clause in SQL SELECT 271
LOG numeric function in SQL 254
LOG10 numeric function in SQL 254
Logical operators 22
Logical operators in SQL 240
Lookup Combo Box creation 199
Lookup Combo Box editing 172
Lookup Combo Box introduced 208
Lookup Combo Box item introduced 154
Lookup Relational InfoViews 199
LOWER CHARCASE clause in SQL ALTER TABLE 295
LOWER CHARCASE clause in SQL CREATE TABLE 289
LOWER string function in SQL 248
LTRIM string function in SQL 248

- M -

Main Menu Bar command 214
Make Hidden button 84
Make Sticky button 84
Many-to-one relationship 208
Margin of Borders and Panels 186
Margins of reports 118
Mark Selection InstaButton 66
Marker color 66
Master-detail relationship 208
Match combo boxes 66
Match Highlight Color 66
MatchBar 66
MatchBar overview for the experts 2
MAX aggregate function in SQL 261
Max summary type 28
MAXIMUM clause in SQL ALTER TABLE 295
MAXIMUM clause in SQL CREATE TABLE 289
Maximum system capacities 227
MDI 19
Memo data type 154
Memo Import/Export 149
Memo item introduced 154
Memo margins 30
Memo operations 30
Memo Ruler 30
Memo Ruler Units 133
Memo Tables 50
Memo Tables explained 30
MemoSearch 66
MemoSearch overview for the experts 2
Menu customization 214
Menus and toolbars general overview 19
Merge Cells button 50
Metafiles 59
Microsoft Access - import from 140
Microsoft Excel - import from 140
Microsoft Office Converters - import/export 149
MIN aggregate function in SQL 261
Min summary type 28
MINIMUM clause in SQL ALTER TABLE 295
MINIMUM clause in SQL CREATE TABLE 289
MINSFROMMSECS function in SQL 267
Mixed InfoView 16

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

MOD arithmetic operator in SQL 240
MOD numeric function in SQL 254
Most Recently Used Symbols 47
Moving columns 22
Moving InfoView Items 163
Moving InfoView Items between Containers 175
MSECSFROMMSECS function in SQL 267
Multi-Line Edit Box item introduced 154
Multilingual support 221
Multiple Document Interface 19
Multiplicity of InstaButtons 214
Multi-user security 97
Multi-user security introduction 94

- N -

Named Styles 74
Naming conventions in SQL 231
Navigation button and dialog 194
NEAR search operator 131
New Field creation 154
New Field Wizard 203
New InfoView creation 160
New InfoView Wizard 203
New record creation 9
New Record Wizard 134
New Table creation 154
New Table Wizard 203
Next DataNav button 16
NOBACKUP clause in SQL ALTER TABLE 295
NOBACKUP clause in SQL OPTIMIZE TABLE 297
NOCASE clause in SQL ALTER TABLE 295
NOCASE clause in SQL CREATE INDEX 294
NOCASE clause in SQL CREATE TABLE 289
NOCASE clause in SQL SELECT 271
NOCHANGE CHARCASE clause in SQL ALTER TABLE 295
NOCHANGE CHARCASE clause in SQL CREATE TABLE 289
NOJOINOPTIMIZE clause in SQL SELECT 271
None color palette 39
NONE COMPRESS clause in SQL ALTER TABLE 295
NONE COMPRESS clause in SQL CREATE INDEX 294
NONE COMPRESS clause in SQL CREATE TABLE 289
Non-Select queries defined 114
Normal hyperlink text color 62
NOT logical operator in SQL 240
NOT NULL clause in SQL ALTER TABLE 295
NOT NULL clause in SQL CREATE TABLE 289
NOT search operator 131
NULLABLE clause in SQL ALTER TABLE 295
NULLABLE clause in SQL CREATE TABLE 289
NULLIF boolean function in SQL 259
Numeric constants in SQL 231
Numeric functions in SQL 254

- O -

Object Class 97
Object Class introduction 94
OCCURS string function in SQL 248
ON clause in SQL CREATE INDEX 294
ON clause in SQL OPTIMIZE TABLE 297
ON clause in SQL SELECT 271
On-Edit Search Updates toggle button 9
One-to-many relationship 208
Open Record Wizard 134
Opening Records in InfoViews 9
Operator list in SQL 231
Operators in search expressions 131
Operators in SQL 240
OPTIMIZE TABLE statement in SQL 297
Options for Alventis 133
Options for bars/menus 214
OR logical operator in SQL 240
OR search operator 131
ORDER BY clause in SQL SELECT 271
Order of Items along the Z axis 163
Ordering columns 22
Outdent button 80
OUTER clause in SQL SELECT 271
Outline bullets and numbering 41
Overview for the experts 2
Overview of Alventis and Designer 4

- P -

Page breaks in reports 118
Page Breaks inserting in Memo 80
Page Control and layout 175
Page Control creation 172
Page Control Style 186

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Panel background 186
Panel creation 172
Panels and layout 175
PanelZoom 19
PanelZoom technical operation 186
Paragraph Alignment 80
Paragraph Background button 39
Paragraph Format InstaButton 36
Parametric queries 116
Parent relationship 175
Password authentication 113
Password Caching 108
Password protection of tables 97
Paste in Memo as HTML button 62
Peer-to-Peer shared access 89
Perform Search button 9
Persistence of interface 19
Personal Styles 81
PI numeric function in SQL 254
Picture (static) creation 172
Picture Format InstaButton 59
Pictures - inserting in Memos 59
PIM - differences from 4
PNG 59
Pointer data type 154
Pointer fields 199
Pointer fields introduced 208
POS string function in SQL 248
POSITION string function in SQL 248
Post DataNav button 16
Posting grid records 22
POWER numeric function in SQL 254
Preview Mode 163
PRIMARY clause in SQL DROP INDEX 302
PRIMARY KEY clause in SQL ALTER TABLE 295
PRIMARY KEY clause in SQL CREATE TABLE 289
Primary keys 208
Print 118
Prior DataNav button 16
Privileges 97
Privileges introduction 94
Proportional Panels 175
Proportional table 50
Proportionality / H button 175
Protection 97
Proximity searches 131

- Q -

Queries List 114
Queries overview 114
Query Form 116
QuickStyle InstaButton 34

- R -

RADIANS numeric function in SQL 254
Radio Button Group caption editing 172
Radio Button Group creation/editing 172
Radio Button Group item introduced 154
RadioGroup Column Count / Currency Precision 186
RAND numeric function in SQL 254
Read-Only item attribute 186
Record and database principles 208
Record creation 9
Record filtering 22
Record grouping 22
Record Hyperlinks 62
Record List grid 118
Record opening 9
Record sorting 22
Record Style Charset 221
Record Styles 81
Records in grids 22
Records of tables defined 208
Rectangle creation 172
REDEFINE clause in SQL ALTER TABLE 295
Redo in Memo 80
Referential integrity 208
Reindex Table button 223
Relational Database principles 208
Relational InfoViews - implementing 199
Remember Current Search button 9
Remove Database command 9
RENAME TABLE statement in SQL 302
Renaming UniGrids 9
Re-ordering columns 22
Repair of data 223
REPAIR TABLE statement in SQL 300
REPEAT string function in SQL 248
Replace in Memos 72
REPLACE string function in SQL 248

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Report 118
Reserved words in SQL 231
Resizing behavior of Containers 175
Resizing columns 22
Resizing InfoView Items 163
Resizing pictures 59
Reveal non-visible characters 49
Reverse Lookup Relational InfoViews 199
Reverse Lookup Relationality introduced 208
Rich Text Format - import/export 149
RIGHT clause in SQL SELECT 271
RIGHT string function in SQL 248
Rights 97
Rights introduction 94
ROLLBACK statement in SQL 303
RotoSplitters 19
ROUND numeric function in SQL 254
Rows in grids 22
Rows of tables defined 208
RTF - import/export 149
RTRIM string function in SQL 248
Ruler 30
Ruler Units 133
RUNSUM aggregate function in SQL 261

- S -

Save InfoView Form button 161
Search Box 9
Search expression 9
Search in multiple tables 4
Search Results pane 9
Search Syntax 131
SECSFROMMSECS function in SQL 267
Security 97
Security overview 94
See-Through 186
Select All button 80
Select Character Set button 221
Select Columns button 50
Select queries defined 114
Select Rows button 50
SELECT statement in SQL 271
Send to Back button 163
Server Administration 108
Server installation and setup 205
Servers grid 89
Sessions of servers 108
Set Charset to Default button in Designer 221
SET clause in SQL UPDATE 283
Settings dialog in Designer 154
Settings for Alventis 133
Setup of the Server 205
Shadow item attribute 186
Shared access 89
Shortcut customization 214
Show Hidden Table Lines button 50
Show Special Symbols button 49
Shrink Height to Shortest button 163
Shrink Width to Narrowest button 163
SIGN numeric function in SQL 254
SIN numeric function in SQL 254
Size (of text) combo box 80
Sizing columns 22
Sizing InfoView Items 163
Small Integer data type 154
Snap to Grid button 163
Sort direction 22
Sorting Arrow 22
Sorting grid records 22
SPACE CHARS clause in SQL ALTER TABLE 295
SPACE CHARS clause in SQL CREATE TABLE 289
Space Equally (H) button 163
Space Equally (V) button 163
Spellcheck 75
Spellcheck Form button 75
SpinEdit Boxes 19
SpinEdit item introduced 154
Split Cells Horizontally button 50
Split Cells Vertically button 50
Split Panel creation 172
Split Panels 175
Splitters 19
Spreadsheets - import from 140
SQL 116
SQL operators 240
SQL Reference overview 230
SQL-92 support overview 230
SQRT numeric function in SQL 254
Stacking of Items along the Z axis 163
START TRANSACTION statement in SQL 302
State Search Results column 9
STDDEV aggregate function in SQL 261
Sticky records 84

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

STOP WORDS clause in SQL ALTER TABLE 295
STOP WORDS clause in SQL CREATE TABLE 289
Stretch of Panels 186
Strikeout button 80
String data type 154
String functions in SQL 248
String operators in SQL 240
Structures of tables 208
Style Charset 221
Style Explorer 85
Style Name column in Search Results 81
Style Name Search Results column 9
Style Picker 74
Styles of records 81
Subject System field 208
Sub-Panels of Split Panels 175
SUBSTRING string function in SQL 248
SUM aggregate function in SQL 261
Sum summary type 28
Summaries of grids 28
Switch InfoView button 16
Symbol Picker 47
Symbol Picker effect on Charset 221
Symbol Picker in Designer 172
Synopsis Search Results column 9
Synopsis Workbench 87
Syntax of search expressions 131
System capacities 227
System Fields intro 4
System fields introduced 208

- T -

Tab Order 194
Tab Order Direct button 194
Tab Sheets of Page Controls 175
Tab Stops 49
Table and database principles 208
Table Cells of Memo tables 50
Table correlation names 231
Table creation/deletion 154
Table Doctor 223
Table Format InstaButton 50
Table formats introduction 4
Table naming conventions in SQL 231
Table protection 97
Table Saving Mode 154
Table structures 208
Table structures introduction 4
Tables Grid in FieldViews 154
Tables in Memos 50
TAN numeric function in SQL 254
Templates of reports 118
Text files - import/export 149
TEXT INDEX clause in SQL ALTER TABLE 295
TEXT INDEX clause in SQL CREATE TABLE 289
Text Search functions in SQL 265
TEXTOCCURS function in SQL 265
TEXTSEARCH function in SQL 265
Thesaurus 75
Tile 186
Time Boxes 19
TIME clause in SQL EXPORT TABLE 298
TIME clause in SQL IMPORT TABLE 299
Time constants in SQL 231
Time data type 154
Time Edit Box item introduced 154
Time Editors format 133
Time operators in SQL 240
TO clause in SQL RENAME TABLE 302
Toggle Client Alignment button 175
Toolbar customization 214
Toolbar style 214
Toolbar style settings 133
ToolTips 133
TOP clause in SQL SELECT 271
TRAILBYTE COMPRESS clause in SQL ALTER TABLE 295
TRAILBYTE COMPRESS clause in SQL CREATE INDEX 294
TRAILBYTE COMPRESS clause in SQL CREATE TABLE 289
Transparency of Panels 186
Transparency of pictures 59
Trashcan button 22
TRIM string function in SQL 248
TRUNC numeric function in SQL 254
TRUNCATE numeric function in SQL 254
Types of data 154

- U -

UCASE string function in SQL 248
Underline button 80
Undo in Memo 80

Copyright © 2002-2007 Alventis Corporation. All rights reserved.

Ungrouping records 22
Unicode 221
Unicode text files - import/export 149
UniGrid 9
UniGrid PanelZoom 9
UniGrid Preview pane 9
UniGrid Search Box 9
UniGrid Search Results pane 9
UniGrid Synopsis column 9
UniGrid Tables grid 9
UNION clause in SQL SELECT 271
UNIQUE clause in SQL ALTER TABLE 295
UNIQUE clause in SQL CREATE INDEX 294
UNIQUE clause in SQL CREATE TABLE 289
UniToggle button 186
Units in Memos 133
Unsupported SQL 239
UPDATE statement in SQL 283
UPGRADE TABLE statement in SQL 301
UPPER CHARCASE clause in SQL ALTER TABLE 295
UPPER CHARCASE clause in SQL CREATE TABLE 289
UPPER string function in SQL 248
URL Hyperlinks 62
Use Hidden button 84
Use Sticky button 84
User Groups 97
User Groups introduction 94
USER MAJOR VERSION clause in SQL ALTER TABLE 295
USER MAJOR VERSION clause in SQL CREATE TABLE 289
USER MINOR VERSION clause in SQL ALTER TABLE 295
USER MINOR VERSION clause in SQL CREATE TABLE 289
Username authentication 113
Username Caching 108
User's Manual organization 4

- V -

Validation 198
Value operators in SQL 240
VALUES clause in SQL INSERT 282
Verification of data 223
Verify / Repair Tables button 223
Version comparison 2
Vertical Shift 33

- W -

Web browsers and non-Latin text Cut-and-Paste 221
Web Hyperlinks 62
WHERE clause in SQL SELECT 271
Width SpinEdit for Item sizing 163
Wildcards in search expressions 131
Windowing in Alventis 19
WITH clause in SQL START TRANSACTION 302
WITH HEADERS clause in SQL EXPORT TABLE 298
WITH HEADERS clause in SQL IMPORT TABLE 299
Wizards in Alventis 134
Wizards in Designer 203
WMF 59
Word data type 154
WORK clause in SQL COMMIT 303
WORK clause in SQL ROLLBACK 303
Workbook - import from 140
Workspace 19

- X -

XML - import from 140

- Y -

YEARSFROMMSECS function in SQL 267

- Z -

Zoomable 186
Z-Order of Items 163

Copyright © 2002-2007 Alventis Corporation. All rights reserved.